SUPPLEMENT TO

Beneath Apple ProDOS

For ProDOS Versions 1.0.1 and 1.0.2

by Don D. Worth and Pieter M. Lechner

QUALITY SOFTWARE
21601 Marilla Street
Chatsworth, California 91311

Apple Books from Quality Software

Beneath Apple DOS $19.95
by Don Worth & Pieter Lechner

Understanding the Apple II $22.95
by Jim Sather

Understanding the Apple IIe (Available Nov. 1984) $24.95
by Jim Sather

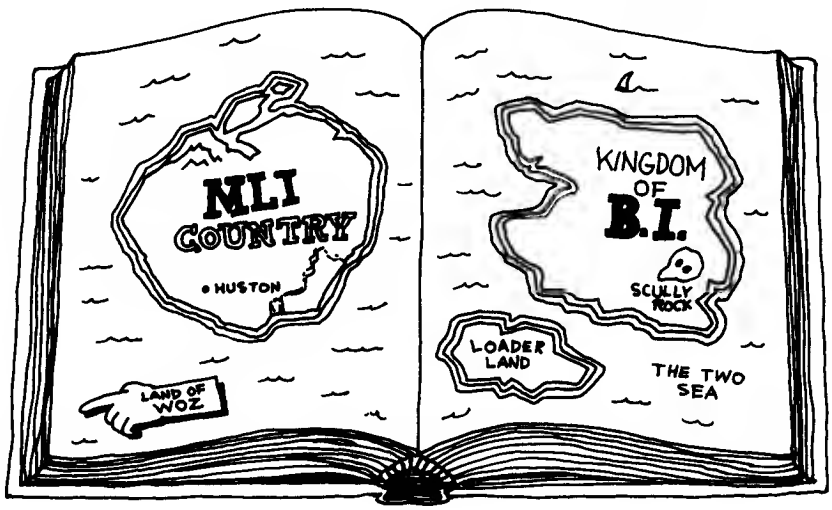
Apple Utility Software from Quality Software

Bag of Tricks (includes diskette) $39.95
by Don Worth & Pieter Lechner

Universal File Conversion (includes diskette) $34.95
by Gary Charpentier

Production Editor: Kathryn M. Schmidt
Illustrations by: George Garcia

CONTENTS



A ProDOS ATLAS

INTRODUCTION

This supplement documents the actual structure and logic of the ProDOS system at nearly a byte by byte level. It is intended to aid experienced programmers in designing customized interfaces to ProDOS, and to provide implicit documentation of ProDOS's functions. Less advanced assembly language programmers may find this supplement useful in learning about how an operating system works. Providing this information does not constitute an endorsement by the authors of indiscriminant modification of the ProDOS components. Whenever possible, standardized interfaces to ProDOS should be used to avoid the uncontrolled modifications which were made to DOS 3.3.

External system programs and utilities such as the FILER and CONVERT are not covered here.

The information provided here is for two releases of the ProDOS operating system--Version 1.Ø.1 and Version 1.Ø.2. Because these versions are so similar, we have included both in the same supplement. Version 1.Ø.1 is first presented in its complete form. Then Version 1.Ø.2 is presented by pointing out and documenting those areas that are different from Version 1.Ø.1.

As new releases of ProDOS become available, additional supplements to Beneath Apple ProDOS will be prepared. To order supplements for other versions of ProDOS, fill out the order form on page 125 of this supplement. Ordering instructions can be found on page 124. When ordering a new supplement, be sure to specify the version of ProDOS you want the new supplement for.

UNDERSTANDING THE LISTINGS

The listings which follow describe the major ProDOS components in great detail. Each module is presented separately and consists of a section defining external addresses referenced by the program (such as zero page usage, I/O select addresses, and global page fields) followed by a section describing the instructions and data in the module. Divisions between major sections and subroutines are indicated with a row of asterisks (*) and additional comments.

Each detail line gives the address of the instruction or data field being described, followed by comments. Within the comments, the following notation is used to indicate references by instructions:

(address)	A store or load reference to a memory or I/O location.
>>address	A branch or jump to an address.
<address>	A call to a subroutine at the indicated address.
-->address	A pointer to an address.

Page titles give the address of the next instruction or data area in the module to be described. These may be used to quickly locate a particular area within the component.

```
Disk Controller Boot ROM -- Apple II/II+/IIe    NEXT OBJECT ADDR: C600
------------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
------------------------------------------------------------------------

C600    MODULE STARTING ADDRESS

        ****************************************
        *                                      *
        *  BOOT ROM - APPLE DISK CONTROLLER    *
        *       THIS CODE RESIDES FROM $C600   *
        *       TO $C6FF, IT LOADS TRACK 0     *
        *       SECTOR 0 INTO RAM AT $800 AND  *
        *       JUMPS TO IT                    *
        *                                      *
        *  VERSION 1.0.1 -- 1 JAN 84           *
        *                                      *
        ****************************************

        ******** ZERO PAGE ADDRESSES ************

0026    SECTOR BUFFER POINTER
002B    SLOT NUMBER * 16 FOR INDEX
003C    WORKBYTE
003D    SECTOR WANTED
0040    TRACK FOUND
0041    TRACK WANTED

        ******** EXTERNAL ADDRESSES *************

0100    SYSTEM STACK
02D6    TRANSLATE TABLE - $80
0300    AUXILIARY BUFFER
0356    TRANSLATE TABLE
0800    SECTORS TO LOAD
0801    ENTRY POINT
C080    PHASE0 OFF
C081    PHASE0 ON
C089    MOTOR ON
C08A    DRIVE SELECT
C08C    READ DATA REGISTER
C08E    SET READ MODE
FCA8    MONITOR WAIT ROUTINE
FF58    RTS

C600 *********** BUILD READ TRANSLATE TABLE **************************

  C600  SIGNATURE
  C602  INITIALIZE TABLE VALUE INDICATOR
  C606  STORE BIT PATTERN
  C609  SHIFT PATTERN LEFT ONE BIT
  C60A  ARE THERE ANY TWO ADJACENT BITS ON?
  C60C  NO, TRY ANOTHER PATTERN  >>C61E
  C60E  YES, TURN OFF RIGHTMOST OF EACH GROUP OF ZEROES
  C610  FLIP BITS, PAIR OF ZERO BITS NOW SINGLE ONE BIT
  C612  HIGH BIT ALWAYS ON/TURN OFF BIT WE MISSED BEFORE
```

```
Disk Controller Boot ROM -- Apple II/II+/IIe    NEXT OBJECT ADDR: C614
------------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
------------------------------------------------------------------------

  C614  --- >>C61E
  C616  SHIFT PATTERN RIGHT, MUST HAVE ONLY ONE BIT ON
  C617  IF MORE THAN ONE BIT ON, TRY ANOTHER PATTERN   >>C614
  C619  FOUND ONE, GET TABLE VALUE
  C61A  AND STORE IT IN TABLE  (0356)
  C61D  INCREMENT TABLE VALUE INDICATOR
  C61E  GET NEXT BIT PATTERN, DONE YET
  C61F  NO, GO CHECK IT OUT  >>C606

C621 *********** DETERMINE SLOT, TURN DRIVE ON ***********************

  C621  CALL A KNOWN RTS  <FF58>
  C624  GET STACK POINTER
  C625  GET HIGH BYTE OF WHERE WE ARE  (0100)
  C628  TIMES 16 TO GET SLOT
  C62C  SAVE SLOT
  C62E  PUT IN X REG FOR INDEX
  C62F  INSURE READ MODE  (C08E)
  C635  SELECT DRIVE 1  (C08A)
  C638  TURN THE MOTOR ON  (C089)

C63B *********** RECALIBRATE DISK ARM *********************************

  C63B  PREPAIR TO STEP THE ARM 80 PHASES
  C63D  TURN A PHASE OFF  (C080)
  C640  PUT COUNTER IN ACCUMULATOR
  C641  CREATE A PHASE NUMBER (0-3)
  C643  DOUBLE IT FOR PROPER INDEX
  C644  COMBINE WITH SLOT FOR FINAL INDEX
  C646  PUT INDEX IN X REGISTER
  C647  TURN A PHASE ON  (C081)
  C64A  DELAY ABOUT 20 MICROSECONDS
  C64F  DECREMENT COUNTER
  C650  LOOP UNTIL ALL 80 ARE DONE  >>C63D

C652 *********** INITIALIZATION ***************************************

  C652  ---
  C654  SECTOR TO FIND -> $00
  C656  TRACK TO FIND -> $00
  C65A  MAIN BUFFER POINTER ($26) -> $0800
  C65C  CLEAR THE CARRY
  C65D  PUSH STATUS ON STACK

C65E *********** SEARCH FOR A VALID HEADER ****************************

  C65E  CHECK DATA REGISTER  (C08C)
  C661  LOOP UNTIL DATA IS VALID  >>C65E
  C663  IS IT A $D5?
  C665  NO, TRY AGAIN  >>C65E
  C667  YES, CHECK REGISTER AGAIN  (C08C)
```

```
Disk Controller Boot ROM -- Apple II/II+/IIe    NEXT OBJECT ADDR: C66A
-------------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------------

  C66A  LOOP UNTIL VALID  >>C667
  C66C  IS IT AN $AA
  C66E  NO, SEE IF ITS A $D5  >>C663
  C670  YES, DELAY FOR REGISTER TO CLEAR
  C671  CHECK REGISTER  (C08C)
  C674  LOOP UNTIL VALID   >>C671
  C676  IS IT A $96
  C678  YES, WE FOUND AN ADDRESS HEADER  >>C683
  C67A  NO, HAVE WE FOUND ONE PREVIOUSLY?
  C67B  IF NOT, START OVER  >>C65C
  C67D  WAS IT AN $AD?
  C67F  YES, WE FOUND A DATA HEADER  >>C6A6
  C681  NO, START OVER  >>C65C

C683 *********** DECODE ADDRESS FIELD *********************************

  C683  INITIALIZE COUNTER
  C685  SAVE VALUE DECODED, WILL BE TRACK ON LAST PASS
  C687  READ DATA REGISTER  (C08C)
  C68A  LOOP UNTIL DATA VALID  >>C687
  C68C  SHIFT BITS INTO POSITION X1X1X1X1
  C68D  SAVE FOR LATER
  C68F  READ REGISTER FOR NEXT BYTE  (C08C)
  C692  LOOP UNTIL VALID  >>C68F
  C694  COMBINE WITH PREVIOUS 1X1X1X1X AND X1X1X1X1
  C696  DECREMENT COUNTER, DONE YET?
  C697  NO, DO ANOTHER  >>C685
  C699  KEEP THE STACK CLEAN
  C69A  IS THIS SECTOR WE WANT?
  C69C  NO, START OVER  >>C65C
  C69E  GET TRACK FOUND
  C6A0  IS IT TRACK WE WANT?
  C6A2  NO, START OVER  >>C65C
  C6A4  YES, INDICATE ADDRESS FOUND, GO LOOK FOR DATA FIELD  >>C65D

C6A6 *********** READ DATA FIELD ************************************

  C6A6  INITIALIZE OFFSET (AUXILIARY BUFFER)
  C6A8  ---
  C6AA  READ DATA REGISTER  (C08C)
  C6AD  LOOP UNTIL VALID  >>C6AA
  C6AF  EXCLUSIVE-OR WITH TRANSLATE TABLE  (02D6)
  C6B4  DECREMENT OFFSET
  C6B5  STORE BYTE IN AUXILIARY BUFFER  (0300)
  C6B8  LOOP UNTIL BUFFER FULL  >>C6A8
  C6BA  INITIALIZE OFFSET (MAIN BUFFER)
  C6BC  READ DATA REGISTER  (C08C)
  C6BF  LOOP UNTIL VALID  >>C6BC
  C6C1  EXCLUSIVE-OR WITH TRANSLATE TABLE  (02D6)
  C6C6  STORE BYTE IN MAIN BUFFER
  C6C8  INCREMENT OFFSET
```

```
Disk Controller Boot ROM -- Apple II/II+/IIe    NEXT OBJECT ADDR: C6C9
-------------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------------

  C6C9  LOOP UNTIL BUFFER FULL  >>C6BA
  C6CB  READ DATA REGISTER  (C08C)
  C6CE  LOOP UNTIL VALID  >>C6CB
  C6D0  IS CHECKSUM OKAY?  (02D6)
  C6D3  NO, START OVER  >>C65C

C6D5 *********** MERGE MAIN AND AUXILIARY BUFFERS**********************

  C6D5  INITIALIZE OFFSET (MAIN BUFFER)
  C6D7  INITIALIZE OFFSET (AUXILIARY BUFFER)
  C6D9  DECREMENT OFFSET (AUX BUFFER)
  C6DA  IF LESS THAN ZERO RESET IT  >>C6D7
  C6DC  GET BYTE FROM MAIN BUFFER
  C6E1  ROLL IN TWO BITS FROM AUXILIARY BUFFER
  C6E6  SAVE COMPLETED DATA BYTE
  C6E8  INCREMENT OFFSET (MAIN BUFFER)
  C6E9  LOOP UNTIL WHOLE BUFFER IS DONE  >>C6D9

C6EB *********** DETERMINE IF THERE IS MORE TO DO*********************

  C6EB  INCREMENT MAIN BUFFER POINTER
  C6ED  INCREMENT SECTOR NUMBER
  C6F1  IS THERE ANOTHER SECTOR TO LOAD?  (0800)
  C6F6  YES, GO DO IT  >>C6D3
  C6F8  NO, ENTER CODE WE JUST LOADED  >>0801

C6FB *********** UNUSED ************************************************

  C6FB  ---
```

```
Oisk Controller Boot ROM -- Apple IIc          NEXT OBJECT AOOR: C600
------------------------------------------------------------------------
AOOR    OESCRIPTION/CONTENTS
------------------------------------------------------------------------

C600    MOOULE STARTING AOORESS

          ****************************************
          *                                      *
          *  BOOT ROM - APPLE //c CONTROLLER ROM *
          *       THIS COOE RESIOES FROM $C600   *
          *       TO $C73F, IT LOAOS TRACK 0     *
          *       SECTOR 0 INTO RAM AT $800 ANO  *
          *       JUMPS TO IT.  IT CAN BOOT FROM *
          *       ORIVE 2 ANO HAS SOME MINIMAL   *
          *       ERROR CHECKING                 *
          *                                      *
          *  VERSION 1.0.1 -- 1 JAN 84           *
          *                                      *
          ****************************************

          ******** ZERO PAGE AOORESSES ************

0003    RETRY COUNT (HIGH BYTE)
0026    SECTOR BUFFER POINTER
002B    SLOT NUMBER * 16 FOR INOEX
003C    WORKBYTE
003O    SECTOR WANTEO
0040    TRACK FOUND
0041    TRACK WANTEO
004F    ORIVE TO BOOT FROM

          ******** EXTERNAL AOORESSES *************

02O6    TRANSLATE TABLE - $80
0300    AUXILIARY BUFFER
0356    TRANSLATE TABLE
07OB    SCREEN LOCATION
0800    SECTORS TO LOAO
0801    ENTRY POINT
C080    PHASE0 OFF
C081    PHASE0 ON
C088    MOTOR OFF
C089    MOTOR ON
C08C    READ OATA REGISTER
C08E    SET REAO MOOE
C0EA    DRIVE SELECT
FCA8    MONITOR WAIT ROUTINE

C600 *********** INITIALIZATION ****************************************

  C600  SIGNATURE
  C602  SET ORIVE -> 1
  C604  INITIALIZE RETRY COUNT (HIGH BYTE)
```

```
Disk Controller Boot ROM -- Apple IIc          NEXT OBJECT AOOR: C606
------------------------------------------------------------------------
AOOR    OESCRIPTION/CONTENTS
------------------------------------------------------------------------

C608 *********** SELECT ORIVE AND TURN IT ON ***************************

  C608  ---
  C60B  INITIALIZE SLOT (6)
  C60O  INITIALIZE OEVICE (1 OR 2)
  C60F  SAVE ORIVE NUMBER ON STACK
  C610  INSURE READ MODE  (C08E)
  C616  GET ORIVE NUMBER BACK
  C617  SELECT APPROPRIATE ORIVE  (C0EA)
  C61A  TURN MOTOR ON  (C089)

C61O *********** RECALIBRATE OISK ARM **********************************

  C61O  PREPAIR TO STEP THE ARM 80 PHASES
  C61F  TURN A PHASE OFF  (C080)
  C622  PUT COUNTER IN A REGISTER
  C623  CREATE A PHASE NUMBER (0-3)
  C625  OOUBLE IT FOR PROPER INOEX
  C626  COMBINE WITH SLOT FOR FINAL INOEX
  C628  PUT INOEX IN X REGISTER
  C629  TURN A PHASE ON  (C081)
  C62C  OELAY ABOUT 20 MICROSECONOS
  C631  OECREMENT COUNTER
  C632  LOOP UNTIL ALL 80 ARE OONE  >>C61F

C634 *********** INITIALIZATION ****************************************

  C634  ---
  C636  SECTOR TO FINO -> $00
  C638  TRACK TO FINO -> $00
  C63A  BUILO THE TRANSLATE TABLE  <C709>

C63O *********** COUNT RETRIES ANO INOICATE ERROR IF BOOT FAILS*******

  C63D  INITIALIZE RETRY COUNT
  C63F  CLEAR THE CARRY
  C640  PUSH STATUS ON STACK
  C641  KEEP STACK CLEAN
  C642  GET SLOT
  C644  DECREMEMT RETRY COUNT, TRY AGAIN?
  C646  YES, GO DO IT  >>C656
  C648  NO, TURN ORIVE OFF  (C088)
  C64B  GET A CHARACTER FROM ERROR MESSAGE  (C6CF)
  C64E  HANG WHEN OONE PRINTING  >>C64E
  C650  PUT A CHARACTER ON THE SCREEN  (077B)
  C653  INCREMENT OFFSET INTO MESSAGE
  C654  GO BACK FOR MORE  >>C64B
  C656  ---
  C657  OECREMENT RETRY COUNT (LOW BYTE)
  C658  IF NOT ZERO, TRY AGAIN  >>C65E
```

```
Disk Controller Boot ROM -- Apple IIc          NEXT OBJECT ADDR: C65A
-------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------

  C65A  IF SO, GO DECREMENT RETRY COUNT (HIGH BYTE)  >>C641
  C65C  SPACE FILLER TO POSITION CODE BELOW  >>C63D

C65E ********** SEARCH FOR A VALID HEADER ***************************

  C65E  CHECK DATA REGISTER  (C08C)
  C661  LOOP UNTIL DATA IS VALID  >>C65E
  C663  IS IT A $D5?
  C665  NO, TRY AGAIN  >>C657
  C667  YES, CHECK REGISTER AGAIN  (C08C)
  C66A  LOOP UNTIL VALID  >>C667
  C66C  IS IT AN $AA
  C66E  NO, SEE IF ITS A $D5  >>C663
  C670  YES, DELAY FOR REGISTER TO CLEAR
  C671  CHECK REGISTER  (C08C)
  C674  LOOP UNTIL VALID  >>C671
  C676  IS IT A $96
  C678  YES, WE FOUND AN ADDRESS HEADER  >>C683
  C67A  NO, HAVE WE FOUND ONE PREVIOUSLY?
  C67B  IF NOT, START OVER  >>C63F
  C67D  WAS IT AN $AD?
  C67F  YES, WE FOUND A DATA HEADER  >>C6A6
  C681  NO, START OVER  >>C63F

C683 ********** DECODE ADDRESS FIELD *********************************

  C683  INITIALIZE COUNTER
  C685  SAVE VALUE DECODED, WILL BE TRACK ON LAST PASS
  C687  READ DATA REGISTER  (C08C)
  C68A  LOOP UNTIL DATA VALID  >>C687
  C68C  SHIFT BITS INTO POSITION X1X1X1X1
  C68D  SAVE FOR LATER
  C68F  READ REGISTER FOR NEXT BYTE  (C08C)
  C692  LOOP UNTIL VALID  >>C68F
  C694  COMBINE WITH PREVIOUS 1X1X1X1X AND X1X1X1X1
  C696  DECREMENT COUNTER, DONE YET?
  C697  NO, DO ANOTHER  >>C685
  C699  KEEP THE STACK CLEAN
  C69A  IS THIS SECTOR WE WANT?
  C69C  NO, START OVER  >>C63F
  C69E  GET TRACK FOUND
  C6A0  IS IT TRACK WE WANT?
  C6A2  NO, START OVER  >>C63F
  C6A4  YES, INDICATE ADDRESS FOUND, GO LOOK FOR DATA FIELD  >>C642

C6A6 ********** READ DATA FIELD ***************************************
```

```
Disk Controller Boot ROM -- Apple IIc          NEXT OBJECT ADDR: C6A6
-------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------

  C6A6  INITIALIZE OFFSET (AUXILIARY BUFFER)
  C6A8  ---
  C6AA  READ DATA REGISTER  (C08C)
  C6AD  LOOP UNTIL VALID  >>C6AA
  C6AF  EXCLUSIVE-OR WITH TRANSLATE TABLE  (02D6)
  C6B4  DECREMENT OFFSET
  C6B5  STORE BYTE IN AUXILIARY BUFFER  (0300)
  C6B8  LOOP UNTIL BUFFER FULL  >>C6A8
  C6BA  INITIALIZE OFFSET (MAIN BUFFER)
  C6BC  READ DATA REGISTER  (C08C)
  C6BF  LOOP UNTIL VALID  >>C6BC
  C6C1  EXCLUSIVE-OR WITH TRANSLATE TABLE  (02D6)
  C6C6  STORE BYTE IN MAIN BUFFER
  C6C8  INCREMENT OFFSET
  C6C9  LOOP UNTIL BUFFER FULL  >>C6BA
  C6CB  READ DATA REGISTER  (C08C)
  C6CE  LOOP UNTIL VALID  >>C6CB
  C6D0  IS CHECKSUM OKAY?  (02D6)
  C6D3  NO, START OVER  >>C6A2

C6D5 ********** MERGE MAIN AND AUXILIARY BUFFERS**********************

  C6D5  INITIALIZE OFFSET (MAIN BUFFER)
  C6D7  INITIALIZE OFFSET (AUXILIARY BUFFER)
  C6D9  DECREMENT OFFSET (AUX BUFFER)
  C6DA  IF LESS THAN ZERO RESET IT  >>C6D7
  C6DC  GET BYTE FROM MAIN BUFFER
  C6E1  ROLL IN TWO BITS FROM AUXILIARY BUFFER
  C6E6  SAVE COMPLETED DATA BYTE
  C6E8  INCREMENT OFFSET (MAIN BUFFER)
  C6E9  LOOP UNTIL WHOLE BUFFER IS DONE  >>C6D9

C6EB ********** DETERMINE IF THERE IS MORE TO DO**********************

  C6EB  INCREMENT MAIN BUFFER POINTER
  C6ED  INCREMENT SECTOR NUMBER
  C6F1  IS THERE ANOTHER SECTOR TO LOAD?  (0800)
  C6F6  YES, GO DO IT  >>C6D3
  C6F8  NO, ENTER CODE WE JUST LOADED  >>0801
  C6FB  JUMP TO DRIVE 2 ENTRY POINT  >>C60B

C6FE ********** UNUSED ************************************************

  C6FE  ---

  C700  MAKE SLOT 7 LOOK EMPTY
  C701  SELECT DEVICE 2
  C703  SELECT DRIVE 2
  C705  SELECT SLOT 6
  C707  GO DO IT  >>C6FB
```

```
Disk Controller Boot ROM -- Apple IIc           NEXT OBJECT ADDR: C707
------------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
------------------------------------------------------------------------


C709 ********** BUILD READ TRANSLATE TABLE **************************

  C709' INITIALIZE BIT PATTERN
  C70B  INITIALIZE TABLE VALUE INDICATOR
  C70D  STORE BIT PATTERN
  C710  SHIFT PATTERN LEFT ONE BIT
  C711  ARE THERE ANY TWO ADJACENT BITS ON?
  C713  NO, TRY ANOTHER PATTERN  >>C725
  C715  YES, TURN OFF RIGHTMOST OF EACH GROUP OF ZEROES
  C717  FLIP BITS, PAIR OF ZERO BITS NOW SINGLE BIT, ETC
  C719  HIGH BIT ALWAYS ON/TURN OFF BIT WE MISSED BEFORE
  C71B  --- >>C725
  C71D  SHIFT PATTERN RIGHT, MUST HAVE ONLY ONE BIT ON
  C71E  IF MORE THAN ONE BIT ON, TRY ANOTHER PATTERN  >>C71B
  C720  FOUND ONE, GET TABLE VALUE
  C721  AND STORE IT IN TABLE  (0356)
  C724  INCREMENT TABLE VALUE INDICATOR
  C725  GET NEXT BIT PATTERN, DONE YET?
  C726  NO, GO CHECK IT OUT  >>C70D
  C728  MAIN BUFFER POINTER ($26) -> $0800
  C72C  INITIALIZE RETRY COUNT (LOW BYTE)
  C72E  RETURN TO CALLER

C72F ********** ASCII ERROR MESSAGE *********************************

C72F
  C72F  "Check        Disk Drive."
  C740  TERMINATE STRING
```

```
ProDOS Loader -- V1.0.1 -- 1 JAN 84              NEXT OBJECT ADDR: 0800
-------------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------------

0800    MODULE STARTING ADDRESS
          ****************************************
          *                                      *
          *  PRODOS LOADER - LOADED FROM SECTORS  *
          *  0 AND 2 OF TRACK 0 (BLOCK 0). LOADER *
          *  LOADS "PRODOS" FILE INTO MEMORY AT   *
          *  AT $2000 AND BRANCHES TO IT.         *
          *  (PRODOS RELOCATOR IS AT $2000)       *
          *                                       *
          *  VERSION 1.0.1 -- 1 JAN 84            *
          *                                       *
          ****************************************

          *** EXTERNAL ADDRESSES ***

0027    ROM BOOT SUBRTN BUFFER PAGE ADDR
002B    ROM BOOT SUBRTN SLOT * 16
003D    ROM BOOT SUBRTN SECTOR TO READ
0040    ROM BOOT SUBRTN CURRENT TRACK
0041    ROM BOOT SUBRTN TRACK TO READ
          -- BLOCK READ PARAMETER LIST --
0042    COMMAND NUMBER (1 = READ)
0043    SLOT * 16
0044    I/O BUFFER ADDRESS ($44/$45)
0045
0046    BLOCK TO READ ($46/$47)
0047
          ------------------------------
0048    POINTER TO BLOCK READ ROUTINE
0049
004A    VOL DIR ENTRY POINTER/FIRST INDEX PAGE
004B
004C    ADDR OF SECOND PAGE OF INDEX BLOCK
004D
004E    INDEX INTO INDEX BLOCK PAGES
0050    TRACK SEEK PHASE-ON INDEX
0051    TRACK PHASE WANTED
0052    BLOCK READER RETRY COUNT
0053    CURRENT TRACK PHASE/PHASE-OFF INDEX
0054
0060    BUFFER POINTER
0061
05AE    SCREEN CENTER LINE
2000    LOAD POINT FOR RELOCATOR
C080    DISK ARM PHASE0
C088    TURN DISK DRIVE OFF
C089    TURN DISK DRIVE ON
FC58    HOME CURSOR/CLEAR SCREEN
```

```
ProDOS Loader -- V1.0.1 -- 1 JAN 84              NEXT OBJECT ADDR: 0800
-------------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------------

          **************************

0800    LOAD UP TO SECTOR 3

0801 ****************** MAIN ENTRY ************************************
            ON ENTRY, X = SLOT*16
                      A = SECTOR NUMBER

  0801   ENTRY POINT
  0802  ALWAYS TAKEN >>0807
  0804  (NEVER EXECUTED) >>A132
  0807  SAVE S0
  0809  READING SECTOR 3 NEXT?
  080B   REMEMBER THIS...
  080D  MAKE $CS FROM SLOT
  0815  AND SAVE AT $49
  0819  $48/49 --> $CSFF IN ROM BOOT
  081C   CHECK $CSFF
  081D  BOOT ROM FOR DISK II?
  081F  NO, HARD DISK BOOT THEN >>085B
  0821  GOT BOTH SECTORS OF LOADER? >>0831
  0823  NO, STOP AT SECTOR 3
  0825  STORE ON PARM (0800)
  0828  SKIP SECTOR 1 (GET SEC 2)
  082A  DUMMY UP $CS5C AS RETURN ADDRESS
  0830   AND CALL ROM SECTOR READ SUBRTN

          ************ LOAD PRODOS ****************
          (ENTIRE LOADER IN MEMORY NOW)

  0831  CURRENT TRACK IS ZERO
  0833  $48/49 --> $CS00
  0837  COPY A PORTION OF DISKETTE BOOT ROM
  0839  TO MY BLOCK READER SUBROUTINE (0994)
  083D  FROM $9F2 TO $A7E
  0843  MODIFY SOME BRANCHES IN THE COPIED CODE (091D)
  0846  TO SUIT MY ERROR HANDLING TASTES (0924)
  084C  AND COPY SECTOR READ SUBROUTINE EXIT CODE (092B)
  084F  TO $A7F TO $A85 (0A7F)
  0855  $48/49 --> DISKETTE BLOCK READER SUBRTN
  0859  AT $0986
  085B  ---
  085D  HARD DISK OR DISKETTE?
  085F  NO, ERROR >>0890
  0861  STORE LSB OF BLOCK READER
  0863  STORE ZEROS IN SEVERAL THINGS
  086E  COMMAND = 1 (READ BLOCK)
  0871  BLOCK NUMBER = 2 (VOL DIRECTORY)
  0875  $60/61 --> $C00 (BUFFER)
  0877  $4A/4B --> $C00 (FIRST ENTRY)
```

```
ProDOS Loader -- V1.0.1 -- 1 JAN 84            NEXT OBJECT ADDR: 0879
-------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------

  0879  READ VOLUME DIRECTORY BLOCKS <0912>
  087C  ERROR? >>08E6
  087E  BUMP TO NEXT BLOCK (2 PAGES)
  0882  NEXT BLOCK NUMBER
  0886  NOW AT BLOCK 6?
  0888  NO, GO READ NEXT ONE >>0879
  088A  YES, CHECK LINK FOR VALIDITY (0C00)
  088D  IT SHOULD BE ZERO FOR VOL DIR (0C01)
  0890  NASTY VOL DIR? >>08FF
  0892  NO, INDEX PAST LINK AND VOL HDR
  0894  AND BEGIN >>0898
  0896  IF ALREADY PROCESSING, USE ENTRY LSB
  0898  ---
  0899  ADD ENTRY LENGTH TO FIND NEXT ENTRY (0C23)
  089D  STILL IN SAME PAGE? >>08AC
  089F  NO, BUMP ENTRY MSB
  08A3  IS IT ODD? (SECOND PAGE OF A BLOCK?)
  08A4  YES... >>08AC
  08A6  NO, JUST FINISHED LAST BLOCK?
  08A8  YES, ERROR -- FILE NOT FOUND >>08FF
  08AA  ELSE, START JUST PAST LINKS
  08AC  UPDATE LSB OF ENTRY POINTER
  08AE  GET NAME LENGTH (0902)
  08B1  TURN OFF FLAGS
  08B4  COMPARE NAME WITH "PRODOS"
  08B9  NOT A MATCH? >>0896
  08BE  IF NAME MATCHES, IS IT A SAPLING FILE?
  08C2  IF NOT, I CAN'T HANDLE IT >>08FF
  08C6  GET FILE TYPE
  08C8  SHOULD BE A PRODOS SYS FILE
  08CA  IF NOT, I GIVE UP >>08FF
  08CD  ALL IS WELL, COPY KEY BLOCK NUMBER
  08CF  TO $46/47
  08D6  $4A/4B AND $60/61 --> $1E00
  08D8  (BUFFER TO HOLD KEY BLOCK)
  08E1  $4C/4D --> $1F00 (SECOND PAGE)
  08E3  READ A BLOCK <0912>
  08E6  ERROR? >>08FF
  08EA  BUMP TO NEXT BLOCK BUFFER
  08EE  $4E = OFFSET INTO INDEX BLOCK
  08F0  GET NEXT BLOCK NUMBER FROM INDEX BLOCK
  08F8  BLOCK NUMBER = 0? (END OF FILE)
  08FA  NOT YET, READ A BLOCK >>08E3
  08FC  ELSE, JUMP TO RELOCATOR AT $2000 >>2000

  08FF  ERROR JUMP >>093F
```

```
ProDOS Loader -- V1.0.1 -- 1 JAN 84            NEXT OBJECT ADDR: 08FF
-------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------


0902 ********** KERNEL NAME ****************************************

  0902  LENGTH OF KERNEL'S NAME
  0903  'PRODOS               ' "PRODOS" (KERNEL NAME)

0912 ********** COPY BLOCK READ BUFFER PTR ***************************

  0912  COPY $60/61 --> $44/45
  0914  (BLOCK READ BUFFER POINTER)
  091A  THEN GO TO BLOCK I/O ROUTINE >>0048

091D ********** ROM SECTOR READ OFFSETS ******************************
            OFFSETS INTO ROM SECTOR READ SUBROUTINE
            TO BRANCH DISPLACEMENTS WHICH NEED TO
            BE CHANGED FOR LOADER'S PURPOSES

  091D  ---

          ******* NEW BRANCH OFFSETS FOR ABOVE ****

  0924  ---

092B ********** SECTOR READ EXIT CODE ********************************
            COPIED INTO DISKETTE SECTOR READ CODE

  092B  GET S0
  092D   AND EXIT NORMALLY
  092E  RETURN
  092F  RESTART BLOCK READ OPERATION >>09BC

0932 ********** 932-93E NOT USED *************************************

  0932  ---

093F ********** ERROR HANDLER ****************************************

  093F  HOME CURSOR/CLEAR SCREEN <FC58>
  0944  COPY "UNABLE TO LOAD PRODOS" MESSAGE (0950)
  0947  TO SCREEN (05AE)
  094D  THEN GO TO SLEEP FOREVER >>094D

  0950  '***           UNABLE TO LOAD PRODOS ***'

096D ********** MOVE ARM TO NEXT PHASE *******************************

  096D  GET CURRENT PHASE
  096F  CONVERT TO NEXT ARM PHASE
  0972  ADD S0
  0975  SELECT NEXT ARM PHASE THIS DRIVE (C080)
  097A  ---
```

```
ProDOS Loader -- V1.0.1 -- 1 JAN 84            NEXT OBJECT ADDR: 097C
-------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------

  097C   DELAY LONG ENOUGH FOR ARM TO MOVE
  0983  WHEN FINISHED, RETURN WITH S0
  0985  RETURN

0986 ********** DISKETTE BLOCK READ ROUTINE **************************
         $44/$45 --> BUFFER
         $46/$47 = BLOCK NO.

  0986  GET BLOCK NO. LSB
  0988  ISOLATE SECTOR REMAINDER
  098C  SKEW SECTOR BY 2
  0992  AND STORE SECTOR WANTED
  0994  GET MSB
  0996  AND HIGH BIT OF TRACK
  0999  MERGE WITH LOW PART OF TRACK
  099C  STORE TRACK WANTED
  099F  TRACK*2 IS PHASE WANTED
  09A3  SET PAGE ADDRESS OF BUFFER
  09A7  TURN DRIVE MOTOR ON (C089)
  09AA  READ SECTOR <09BC>
  09AD  NEXT PAGE
  09B1  SKEW TO NEXT SECTOR
  09B5  READ SECOND SECTOR OF BLOCK <09BC>
  09B8  THEN TURN MOTOR OFF AND EXIT (C088)
  09BB  RETURN

        ******* DISKETTE SECTOR READ ROUTINE ****

  09BC  GET CURRENT TRACK
  09BF  CONVERT TO PHASE
  09C5  GET CURRENT PHASE
  09C7  STORE FOR PHASE OFF
  09CA  SUBTRACT PHASE WANTED TO DETERMINE...
  09CC  DIRECTION -- ON CORRECT TRACK NOW? >>09E2
  09D0  NO, ADJUST PHASE UP...
  09D4  OR DOWN AND...
  09D6  ---
  09D7  SEEK ARM ONE PHASE... <096D>
  09DD  IN PROPER DIRECTION <096F>
  09E0  UNTIL WE ARE THERE >>09C5
  09E2  ---
  09E4  RETRY COUNT OF 127
  09E7  ---
  09E9  LOWER RETRY COUNT
  09EB  RETRIES EXHAUSTED? >>09BB
  09EF   RETRIES FOR A $D5 HEADER
```

```
ProDOS Loader -- V1.0.1 -- 1 JAN 84            NEXT OBJECT ADDR: 09F0
-------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------

        ******* DEVICE DEPENDENT SECTOR READ ****
          COPIED FROM ROM ON DISKETTE CARD
          SEE $CX5E IN BOOT ROM

  09F2    START OF SECTOR READ ROUTINE
  0A7F    BASE ADDR FOR MODIFICATIONS

09F2 ********** A86-BFF NOT USED ***********************************

09F2 ********** VOLUME DIRECTORY BUFFER ****************************

  0C00    START OF VOLUME DIRECTORY BUFFER
  0C01
  0C23    OFFSET TO ENTRY LENGTH FIELD
```

```
ProDOS Relocator -- V1.0.1 -- 1 JAN 84          NEXT OBJECT ADDR: 2000
-------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------

2000    MODULE STARTING ADDRESS


        ********************************
        *                              *
        *  PRODOS RELOCATOR            *
        *       LOADED AS THE FIRST    *
        *       PORTION OF THE PRODOS  *
        *       IMAGE AT $2000.        *
        *                              *
        *  VERSION 1.0.1 -- 1 JAN 84   *
        *                              *
        ********************************

          ******** ZERO PAGE ADDRESSES ************

000A    AUTOSTART ROM CHECKSUM POINTER
000B
000C    CONFIGURATION BYTE (MACHID TO BE)
0010    GENERAL PURPOSE POINTER
0011
0012    DISK TYPE (0=DISK II, 4=PROFILE)
0013     AND INPUT RELOC RANGE POINTER
0014    VOL DIR ENTRY POINTER FOR RELOCATOR
0015     AND OUTPUT RANGE PTR
0016    LENGTH OF RELOCATION RANGE
0017
0018    INPUT RELOCATION RANGE POINTER
0019
001A    END OF INPUT RANGE
001B
003C    GENERAL PURPOSE POINTER
003D
003E    GENERAL PURPOSE POINTER
003F
0040    RAMDRIVE OUTPUT POINTER
0041
0042    VARIOUS USES: PARM TO AUXMOVE,
0043     UNIT/SLOT PASSED TO RELOCATOR
0046    BLOCK NUMBER TO RAMDRIVE
0047


          ******** EXTERNAL ADDRESSES *************

0080    MACHID BUILD SUBRTN FOR 128K

0280    GENERAL PURPOSE BUFFER
0281    BUFFER+1
```

```
ProDOS Relocator -- V1.0.1 -- 1 JAN 84          NEXT OBJECT ADDR: 2000
-------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------

          ******** SCREEN LINE ADDRESSES **********

04B6    SCREEN BUFFER LINE
05A9    SCREEN BUFFER LINE
05AE    SCREEN BUFFER LINE
06B3    SCREEN BUFFER LINE
07A8    SCREEN BUFFER LINE
07AD    SCREEN BUFFER LINE
07D0    SCREEN BUFFER LINE

          ******** INTERP LOADER ADDRESSES ********

0800    ENTRY OF INTERP LOADER
08E2    'UNABLE TO FIND SYSTEM FILE'
090A    'INTERP FILE TOO LARGE'
092A    'UNABLE TO LOAD ...'
093B    INTERP FILE NAME ITSELF
093C    +1
094F    LENGTH OF MESSAGE
0950    MLI: OPEN LIST
0956    MLI: GET EOF
0958    EOF MARK
0959    EOF MARK+1
095A    EOF MARK+2 (MSB)
095B    MLI: READ LIST
095F    READ BUFFER ADDR
0960    +1
0963    MLI: CLOSE LIST
0965    '.SYSTEM'

0C00    VOLUME DIRECTORY BUFFER
0C23    ENTRY LENGTH
          -- RAMDRIVE VOLUME DIRECTORY
0E04    VOLUME HDR, VOLUME NAME
0E22    VOLUME HDR, ACCESS-TOTAL BLOCKS

          ******** SYSTEM GLOBAL PAGE *************

BF00    ENTRY POINT FOR MLI
BF03    QUIT VECTOR
BF06    DATE/TIME
BF10    DEVICE HANDLER TABLES
BF30    LAST DEVICE USED
BF31    NUMBER OF ACTIVE DISK DEVICES
BF32    ACTIVE DISKS SEARCH LIST
BF98    MACHINE TYPE FLAGS
BF99    SLOT WHICH CONTAIN CARDS WITH ROM
BFFF    TOP OF 48K RAM
```

```
ProDOS Relocator -- V1.0.1 -- 1 JAN 84          NEXT OBJECT ADDR: 2000
-------------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------------

        ******** I/O PORT ADDRESSES *************

C000    80 STORE OFF
C001    80 STORE ON
C002    READ MAIN RAM
C003    READ AUX RAM
C004    WRITE MAIN RAM
C005    WRITE AUX RAM
C008    MAIN STACK/ZERO PAGE
C009    ALTERNATE STACK/ZERO PAGE
C00C    80 COLUMN DISPLAY OFF
C018    READ 80STORE SWITCH
C030    SPEAKER
C082    MOTHERBOARD ROM READ ENABLE
C083    READ/WRITE RAM 2ND 4K BANK
C08B    READ/WRITE RAM 1ST 4K BANK
C311    MOVE TO/FROM AUXMEM SUBROUTINE
C314    XFTER TO AUXMEM SUBRDUTINE
CFFF    RESET I/O CARD ROMS

        ******** RAM CARD ADDRESSES *************

D000    KERNEL START (APPLESOFT START)
FF00    START OF DEVICE DRIVERS

        ******** MONITOR ROM ********************

FB1E    PADDLE READ SUBROUTINE
FB2F    MONITOR INIT ROUTINE
FBB3    ROM VERSION BYTE
FBC0    SECONDARY VERSION BYTE (0-3)
FC58    CLEAR SCREEN
FE84    SET NORMAL VIDEO
FE89    IN#0
FE93    PR#0


2000 *********** PRODOS RELOCATOR MAIN ENTRY *************************

  2000  STORE SLOT IN MLI ONLINE PARMS
  2005  PRINT "APPLE II PRODOS..." <2448>
  200E  MOVE 3PAGE/INTERP LOADER ETC. <2663>
  2011  NO ERROR? >>2016
  2013  ERROR >>212A
  2016  ---
  201A  THERE MUST BE A MINIMUM OF 48K (BFFF)
  2021  IF NOT, ERROR >>2097
  2029  MAKE DOUBLY SURE >>2097
  202B  SELECT MOTHERBOARD ROMS (C082)
```

```
ProDDS Relocator -- V1.0.1 -- 1 JAN 84          NEXT OBJECT ADDR: 202E
-------------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------------

  202E  DETERMINE MACHINE TYPE <23B0>
  2031  PICK UP CDNFIGURATION BYTE
  2033  64K DR MORE MEMORY?
  2035  YES, GDT A LANGUAGE CARD >>2047
  2037  ELSE,
  2039  USE THE 48K RELOCATION TABLES (2179)
  203C  INSTEAD OF THE 64K ONES (2173)
  2042  AND PUT DISK DRIVERS AT $B800

        ******** RELOCATE PRDDDS ***************

  2047  ---
  204D  COPY/RELOCATE PRODOS ITSELF <2663>
  2050  ERROR? >>2097
  2052  ENABLE MOTHERBOARD ROMS AGAIN (C082)
  2055  GET SECONDARY MACHINE TYPE (FBC0)
  2058  MUST BE 0 THRU 3
  205A  ELSE, IGNORE IT >>2076
  205C  GET MACHID
  205E  INDICATE "FUTURE SYSTEM"
  2060  TURN OFF MACHINE INDICATORS
  2062  $FBC0 = $00 ?
  2064  NO >>2068
  2066  YES, MACHID=$80
  2068  $FBC0 = $01 ?
  206A  ND >>206E
  206C  YES, MACHID=$40
  206E  $FBC0 = $03 ?
  2070  NO, MACHID =$00 >>2074
  2072  YES, MACHID=$C0
  2074  REPLACE UPDATED MACHID
  2076  COPY BOOT DEVICE ID TO READ BLDCK PARMS (2165)
  207C  AND AS LAST DEVICE USED (BF30)
  207F  DETERMINE PERIPHERAL CARD CONFIGURATIDN <24E6>
  2082  BOOT DEVICE TO... (216C)
  2085  GLOBAL PAGE LAST DEVICE USED (BF30)
  208B  WRITE ENABLE 1ST BANK DF LANG. CARD (C08B)
  2094  COPY CLOCK CODE TO DEVICE DRIVER AREA <2663>
  2097  ERROR? >>20C6
  2099  CHECK MACHINE TYPE AGAIN (BF98)
  209C  GOT 64K OR MORE?
  20A0  NO >>20C8
  20A2  YES, QUIT VECTOR --> $EECF
  20AC  OPEN L.C. FOR WRITE, 1ST 4K (C083)
  20B5  POINT TO QUIT CODE TABLE (2178)
  20B8  MOVE QUIT CODE TO L.C. <2663>
  20BD  STORE QUIT VECTOR START PAGE (D000)
  20C0  OPEN L.C. FOR WRITE, 2ND 4K... (C08B)
  20C3  AGAIN (C08B)
  20C6  ERROR DURING MOVE? >>212A
  20C8  GET MACHID YET AGAIN (BF98)
```

```
ProOOS Relocator -- Vl.0.1 -- 1 JAN 84          NEXT OBJECT ADDR: 20CB
-------------------------------------------------------------------
ADOR    OESCRIPTION/CONTENTS
-------------------------------------------------------------------

  20CB  128K?
  20CF  NO... >>2004
  2001  YES, ESTABLISH RAM ORIVE IN UPPER 64K <28FF>

      ******** GET VOL LABEL ******************

  2004  MLI: ONLINE OEVICE CALL <BF00>
  200A  ERROR? >>212A      /
  200F  VALIO VOLUME NAME?
  20E1  IF NOT, ERROR >>212A
  20E4  ELSE, BUMP LENGTH BY ONE
  20E9  ANO PREFIX NAME BY A "/"
  20EE  MLI: SET PREFIX <BF00>
  20F4  ERROR? >>212A

        ******** REAO VOLUME OIRECTORY **********

  20F6  ---
  20F8  $14/15 --> $C00
  20FE  ---
  2103  BLOCK = 2 (VOLUME OIRECTORY) (216F)
  2109  MLI: REAO BLOCK <BF00>
  210F  ERROR? >>212A
  2113  GET NEXT BLOCK NUMBER
  2119  IF ZERO, END OF VOLUME OIRECTORY >>2127
  2121  AOO TWO PAGES (ONE BLOCK) TO POINTER
  2123  AND STOP AT $1400 IN ANY CASE
  2125  ELSE, REAO NEXT BLOCK AS WELL >>20FE
  2127  WHEN OONE, JUMP TO INTERP LOAOER >>0800

212A ********** ERROR HANDLER ************************************

  212A  ENABLE MOTHERBOARD ROMS (C082)
  212D  CLEAR SCREEN <FC58>
  2132  PRINT "RELOCATION/CONFIG ERROR" (213E)
  213B  THEN SLEEP FOREVER >>213B

213E ********** DATA ************************************************

  213E  ---
  213E  '**           RELOCATION / CONFIGURATION ERROR **'

  2164  MLI: ONLINE PARMS
  2165          SLOT*16 ANO ORIVE
  2166          REAO THEM TO $281

  2168  MLI: SET PREFIX PARMS
  2169          PREFIX IS AT $280
```

```
ProOOS Relocator -- Vl.0.1 -- 1 JAN 84          NEXT OBJECT AOOR: 2169
-------------------------------------------------------------------
AOOR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------

  216B  MLI: REAO BLOCK PARMS
  216C          DEVICE
  216O          BUFFER
  216F          BLOCK NUMBER

  2171  AOORESSES OF RELOCATION TABLES

2170 ********** RELOCATION TABLES **********************************
          +0:     00 - ZERO BLOCK OF MEMORY
                  01 - COPY BLOCK
                  02 - RELOCATE MSB AOORESSES
                  03 - RELOCATE 2 BYTE AOORS
                  04 - RELOCATE INSTRUCTIONS
          +1/2:   AODR OF OUTPUT BLOCK
          +3/4:   LENGTH OF BLOCK IN BYTES
          +5/6:   AODR OF INPUT BLOCK (IF ANY)
          +7:     NUM RANGES TO CORRECT FOR (-1)
          +8:             START PAGES
          +8+COUNT:       END PAGE ADDRESSES
          +8+COUNT+COUNT:AODITIVE CORRECTION FACTOR

          ******** COMMON MOVES TABLE *************

  2170  COPY (INTERPRETOR LOAOER)
  217E          TO =$800
  2180          LEN=$16C
  2182          FRM=$2234
  2184  COPY (3 PAGE IMAGE)
  2185          TO =$3F0
  2187          LEN=$10
  2189          FRM=$23A0
  218B  COPY (CHECKSUM)
  218C          TO =$0A
  218E          LEN=$02
  2190          FRM=$14
  2192  COPY (RAM DRIVE BANK SWITCHER)
  2193          TO =$80
  2195          LEN=$47
  2197          FRM=$2401
  2199  ENO OF TABLE

        ******** QUIT COOE MOVE TABLE ***********

  219A  COPY (QUIT COOE)
  219B          TO =$D100
  219D          LEN=$300
  219F          FRM=$5900
  21A1  END OF TABLE
```

```
ProDOS Relocator -- V1.0.1 -- 1 JAN 84          NEXT OBJECT ADDR: 21A1
-------------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------------

        ******** 48K PRODOS RELOC TABLE *********

21A2  COPY (SYSTEM GLOBAL PAGE IMAGE)
21A3          TO =$BF00
21A5          LEN=$100
21A7          FRM=$4F00
21A9  ZERO (PRODOS KERNEL DATA AREA)
21AA          ADR=$B100
21AC          LEN=$700
21AE  COPY (PRODOS KERNEL)
21AF          TO =$9000
21B1          LEN=$2100
21B3          FRM=$2D00
21B5  RELOCATE INSTRUCTIONS
21B6          TO =$9000
21B8          LEN=$1ECE
21BA          FRM=$9000
21BC          FOR ADDRS=$D0XX-$F7XX
21BF          ADJUST BY=$C0
21C0  RELOCATE ADDRESSES
21C1          TO =$AF65
21C3          LEN=$28
21C5          FRM=$AF65
21C7          FOR ADDRS=$D0XX-$F0XX
21CA          ADJUST BY=$C0
21CB  COPY (DEVICE DRIVERS)
21CC          TO =$B800
21CE          LEN=$700
21D0          FRM=$5200
21D2  RELOCATE INSTRUCTIONS
21D3          TO =$B800
21D5          LEN=$195
21D7          FRM=$B800
21D9          FOR ADDRS=$F8XX-$FEXX
21DC          ADJUST BY=$C0
21DD  RELOCATE INSTRUCTIONS
21DE          TO =$BB85
21E0          LEN=$339
21E2          FRM=$BB85
21E4          FOR ADDRS=$F8XX-$FEXX
21E7          ADJUST BY=$C0
21E8  END OF TABLE

        ******** 48K PRODOS CLOCK TABLE *********

21E9  COPY (CLOCK DRIVER)
21EA          TO =$B142
21EC          LEN=$7D
21EE          FRM=$5000
21F0  RELOCATE INSTRUCTIONS
```

```
ProDOS Relocator -- V1.0.1 -- 1 JAN 84          NEXT OBJECT ADDR: 21F1
-------------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------------

21F1          TO =$B142
21F3          LEN=$69
21F5          FRM=$B142
21F7          FOR ADDRS=$C1-C1
21FA                   =$F1-F1
21FC          ADJUST BY=$S0 AND $C0
21FE  END OF TABLE

        ******** 64K PRODOS RELOC TABLE *********

21FF  COPY (INTERRUPT VECTORS)
2200          TO =$FF80
2202          LEN=$80
2204          FRM=$5080
2206  COPY (SYSTEM GLOBAL PAGE)
2207          TO =$BF00
2209          LEN=$100
220B          FRM=$4E00
220D  ZERO (PRODOS KERNEL DATA AREA)
220E          ADR=$F100
2210          LEN=$700
2212  COPY (PRODOS KERNEL)
2213          TO =$D000
2215          LEN=$2100
2217          FRM=$2D00
2219  COPY (DEVICE DRIVERS)
221A          TO =$F800
221C          LEN=$700
221E          FRM=$5200
2220  END OF TABLE

        ******** 64K PRODOS CLOCK TABLE *********

2221  COPY (CLOCK CODE)
2222          TO =$F142
2224          LEN=$7D
2226          FRM=$5000
2228  RELOCATE INSTRUCTIONS
2229          TO =$F142
222B          LEN=$69
222D          FRM=$F142
222F          FOR ADDRS=$C1XX-$C1XX
2232          ADJUST BY=$S0
2233  END OF TABLE
```

```
ProDOS Relocator -- V1.0.1 -- 1 JAN 84          NEXT OBJECT ADDR: 2234
-------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------

2234 *********** INTERPRETER LDADER *********************************
      '    (LDADED AT $2234, MOVED TO $800)

  2234  $10/11 --> VOLUME DIRECTOR ENTRIES
  2236  INITIALLY AT $C00
  2238  DFFSET BEYOND LINKS (+4)
  223A  (TURN NEXT INSTRUCTION INTD BIT)

        ******** SCAN DIRECTORY FOR INTERP ******

  223B  PICK UP LSB
  223E  BUMP BY ENTRY LENGTH (0C23)
  2241  UPDATE LSB
  2243  PAGE OVERFLOW? >>2257
  2245  NO, ROOM FOR ONE MORE ENTRY? (0C23)
  224A  NO, CHECK MSB
  224D  START OF A BLOCK? >>2259
  224F  NO, AT END OF DIRECTORY?
  2251  YES, FILE NOT FOUND IN DIRECTORY >>2271
  2253  NO, START NEW BLOCK AT +4
  2255  AND UPDATE LSB
  2257  BUMP MSB
  2259  ---
  225D  CHECK FILE TYPE FOR PRODDS "SYS" FILE
  225F  NOT IT? >>223B
  2262  INACTIVE ENTRY?
  2264  IF SO, SKIP IT >>223B
  2268  SAVE NAME LENGTH AT $280 (0280)
  226D  MUST BE AT LEAST 8 CHARS LDNG >>223B
  226F  JUMP AROUND ERROR CODE >>2274
  2271  ERROR - INTERP FILE NOT FOUND >>22E1
  2273   HARD BREAK IN THAT CASE
  2274  ---
  2277  IS THIS ".SYSTEM"?
  2279  (SEE $2399) (0965)
  227D  NO, SKIP ENTRY >>223B
  2281  CHECK ALL CHARACTERS IN NAME >>2277

        ******** LOAD INTERPRETER AT $2000 ******

  2283  ---
  2285  ---
  2286  COPY NAME TO $281
  228D  AND TO "UNABLE TO LOAD" MSG (093B)
  2295  ADD BLANK AT END OF NAME
  2297  IN MESSAGE (093C)
  229B  NAMELEN + ERRORMSGLEN
  229D  SAVE AT $2383 (094F)
  22A0  MLI: OPEN .SYSTEM FILE <BF00>
  22A4  (PARM LIST AT $2384)
```

```
ProDOS Relocator -- V1.0.1 -- 1 JAN 84          NEXT OBJECT ADDR: 22A6
-------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------

  22A6  ERRDR? >>22EE
  22A8  MLI: GETEOF <BF00>
  22AC  (PARM LIST AT $238A)
  22AE  ERRDR? >>22EE
  22B0  GET MSB (SEE $238E) (095A)
  22B3  BIGGER THAN 64K???? >>2308
  22B8  MUST BE LESS THAN $9800 BYTES
  22BA  OR ERROR...  >>2308
  22BC  STDRE LENGTH IN MLI READ LIST (0960)
  22C2  AND LSB TDD (095F)
  22C5  MLI: READ INTERPRETER INTO $2000 <BF00>
  22C9  (PARM LIST AT $238F)
  22CB  NO ERRORS? >>22D3
  22CD  ERROR, BAD BUFFER?
  22CF  YES, FILE WAS TOO LARGE >>2308
  22D1  ELSE, "UNABLE TO LOAD ..." >>22EE
  22D3  MLI: CLOSE INTERPRETER FILE <BF00>
  22D7  (PARM LIST AT $2397)
  22D9  ERROR? >>22EE
  22DB  NO, ENABLE MOTHERBOARD ROMS (C082)
  22DE  AND JUMP TO INTERPRETER >>2000

22E1 *********** ERROR HANDLERS *************************************

  22E1  ---
  22E3  PRINT "UNABLE TD FIND A .SYSTEM FILE" (08E2)
  22EC  THEN GO TO SLEEP >>2313

  22EE  GET NAME LENGTH (094F)
  22F1  LINE LENGTH
  22F4  LESS NAME LENGTH (094F)
  22F7  DIVIDED BY 2
  22F8  GIVES OFFSET TO CENTER THE LINE (094F)
  22FC  PRINT "UNABLE TO LOAD ..." (092A)
  2306  GO TO SLEEP FOREVER >>2313

  2308  ---
  230A  PRINT "SYSTEM PROGRAM TOO LARGE" (090A)
  2313  GO TO SLEEP FOREVER >>2313

2316 *********** DATA AREA ****************************************

  2316  '**             UNABLE TO FIND A ".SYSTEM" FILE  **
  233E  '**             SYSTEM PROGRAM TOO LARGE  **'
  235E  '**            UNABLE TO LOAD X.SYSTEM **********'
  2383  NAME LEN +13H (LEN OF MSG)
```

```
ProDOS Relocator -- V1.0.1 -- 1 JAN 84          NEXT OBJECT ADDR: 2383
-------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------

  2384  MLI: OPEN PARM LIST
  2385  PATHNAME IS AT $280
  2387  I/O BUFFER AT $1400
  2389  REFNUM=1

  238A  MLI: GET EOF PARM LIST
  238B  REFNUM=1
  238C  EOF MARK POSITION

  238F  MLI: READ LIST
  2390  REFNUM=1
  2391  READ TO $2000
  2393  LENGTH (FROM EOF MARK)
  2395  ACTUAL LENGTH READ

  2397  MLI: CLOSE LIST
  2398  REFNUM=0, CLOSE ALL FILES

  2399  '.SYSTEM'

23A0 *********** END OF INTERP LOADER ****************************

23A0 *********** 3 PAGE VECTOR IMAGE *********************************

  23A0  BRK HANDLER AT $FA59
  23A2  RESET AT $FF59
  23A4  POWER UP BYTE
  23A5  & VECTOR TO $FF59 >>FF59
  23A8  CTL-Y VECTOR TO $FF59 >>FF59
  23AB  NMI VECTOR TO $FF59 >>FF59
  23AE  IRQ HANDLER AT $BFEB (PRODOS)


23B0 *********** DETERMINE MACHINE ID *********************************
                $0C=00.. .... APPLE II
                    01.. .... APPLE II+
                    10.. .... APPLE IIE
                    11.. .... APPLE /// EMULAT.
                    ..01 .... 48K RAM
                    ..10 .... 64K RAM
                    ..11 .... 128K RAM
                    .... 0... CURRENT MACHINE
                    .... 1... FUTURE MACHINE
                    .... ..1. 80 COL CARD
                    .... ...1 THUNDER CLOCK
```

```
ProDOS Relocator -- V1.0.1 -- 1 JAN 84          NEXT OBJECT ADDR: 23B0
-------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------

  23B0  ASSUME NOTHING AT FIRST
  23B4  GET A ROM BYTE (FBB3)
  23B7  APPLE II?
  23B9  YES, SET BIT >>23DA
  23BB  NO,
  23BD  APPLE IIE?
  23BF  YES, SET BIT >>23DA
  23C1  NO,
  23C3  APPLE II+?
  23C5  NO, WHAT IS IT? >>23D4
  23CA  REALLY A II+?
  23CC  YES >>23DA
  23D0  /// EMULATION MODE?
  23D2  YES  >>23FE
  23D4  OTHERWISE, UNKNOWN MACHINE
  23D6  CREATE INVALID INSTR AT $80
  23D8  AND GO THERE >>23FE
  23DA  UPDATE MACHID
  23DF  READ/WRITE ENABLE 1ST BANK LANG CARD (C08B)
  23E4  CHECK FOR L.C. EXISTANCE (D000)
  23F6  IF PRESENT, MARK IN MACHID
  23FA  ELSE, ONLY 48K AVAILABLE
  23FC  ADD TO MACHID
  23FE  THEN GO TO $80 (CODE BELOW) >>0080

2401 *********** LOOK FOR EXTENDED 80 COL CARD ***********************
         (CODE MOVED TO $80 TO ALLOW BANK SWITCH)

  2401  UPDATE MACHID
  2403  IIE? >>243A
  2405  YES,
  2407  BANK TO AUX MEMORY (C005)
  240D  STORE A PATTERN THERE (0C00)
  2416  MAKE SURE PATTERN STAYS THERE
  2428  ---
  2429  IF NOT, ONLY GOT 64K TOTAL >>242C
  242B   ELSE, GOT 128K TOTAL
  242C  BANK BACK TO MAIN MEMORY (C004)
  2432  64K? >>243A
  2436  NO, INDICATE 128K
  2438  IN MACHID
  243A  SET UP $A/B --> "APPLE II"
  243D  IN AUTOSTART MONITOR ROM
  243F  AT $FB09
  2441  BUT DO IT IN A CONVOLUTED WAY
  2447   RETURN TO CALLER
```

```
ProDOS Relocator -- V1.0.1 -- 1 JAN 84          NEXT OBJECT ADDR: 2447
-------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------

2448 *********** DISPLAY LOAD MESSAGE *********************************

  2448  CLICK SPEAKER  (C030)
  244B  STORE IN MAIN MEMORY  (C00C)
  244E  80 COL DISPLAY OFF (C000)
  2451  SET NORMAL VIOEO <FE84>
  2454  CALL MONITOR INITIALIZATION <FB2F>
  2457  SET VIOEO PR#0 <FE93>
  245A  SET KEYBD IN#0 <FE89>
  2450   OUT OF OECIMAL MOOE
  245E   DISABLE FOR INTERRUPTS
  245F  CLEAR SCREEN <FC58>
  2464  PRINT "APPLE II" (2492)
  246F  PRINT "PROOOS 1.0.1 ETC." (249A)
  247A  PRINT BLANKS (24B1)
  2485  PRINT "COPYRIGHT ETC." (24BF)
  248E  CLICK SPEAKER AGAIN (C030)
  2491   OONE

2492 *********** OATA AREA *****************************************

  2492  'APPLE         ][' 
  249A  'PROOOS        1.0.1   1-JAN-84'
  24B1  '                              '
  24BF  'COPYRIGHT    APPLE COMPUTER, INC., 1983-84'


24E6 *********** OETERMINE SLOT CONFIGURATION ************************

  24E6  ---
  24E8  ZERO SOME THINGS
  24EF  NO OISKS ACTIVE YET (BF31)
  24F4  $10/11 --> $C700 (LOOP THRU ALL SLOTS)
  24F6  RESET I/O CARO ROMS (CFFF)
  24FB  CHECK SIGNATURE ON CARD FOR DISK DEVICE
  2501  NOT OISK? >>2569
  2507  GET $CSFF BYTE (TYPE OF DISK)
  2509  OISK II? >>252B
  250B  NO, PROFILE?
  250D  NO? THEN NOT A OISK >>2569

          ******** PROFILE FOUND ******************

  250F   ELSE, SAVE AS LSB OF BLOCK READ SUBRTN
  2511  GET $CSFE (STATUS BYTE)
  2514  CAN WE AT LEAST READ STATUS AND DATA?
  2518  YES? >>251F
  251A   NO,
  251D  NOT A OISK AFTER ALL >>2569
  251F   GET STATUS BYTE AGAIN
```

```
ProDOS Relocator -- V1.0.1 -- 1 JAN 84          NEXT OBJECT AOOR: 2523
-------------------------------------------------------------------
AOOR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------

  2523  TOP NIBBLE IS DEVICE ID
  2524  PROFILE SHOULO BE $04
  2526  CHECK NUMBER OF VOLS (SHOULD BE 0)
  2527  GET SLOT NO. FOR OEVICE ORIVER LOC.
  2529  AND GO DO COMMON PROCESSING FOR DISK >>2535

          ******** OISK II FOUNO ******************

  252B  $12 ZERO FOR OISK II
  2520  GET DISK II DEVICE DRIVER LOCATION (2627)
  2531  ($F800 OR $B800) (2628)
  2534   DISK II HAS 2 ORIVES

          ******** OISK FOUNO **********************

  2535   SAVE DEVICE AODRESS
  2537  SET UP INOEX OF SLOT*2
  253F  BUILD ST (S=SLOT, T=0 OISKII,4 PROFILE)
  2542  BUMP DEVICE COUNT BY ONE (BF31)
  2546  ANO ADD ORIVE TO SYSTEM SEARCH LIST (BF32)
  254A   NUMBER OF ORIVES
  254C  ONLY ONE? >>2552
  254E   NO, BUMP INOEX
  254F  ANO MARK SECOND ORIVE IN SEARCH LIST (BF32)
  2552  STORE FINAL OEVICE COUNT (BF31)
  2557  SET UP DISK OEVICE ORIVER VECTORS (BF11)
  255A  IN SYSTEM GLOBAL PAGE >>2564
  255C  (SET UP TWO VECTORS FOR A OISK II) (BF21)
  2564  ---
  2568   I RECOGNIZE THIS CARO
  2569  GO MARK SLTBYT TO SHOW ROMS IN SLOT <25B6>
  2570  OO ALL CAROS EXCEPT
  2572  SLOT 0 ($C000) >>24F6
  2578  GET LAST OISK OEVICE IN SEARCH LIST (BF32)
  257E  BOOT ORIVE? (BF30)
  2582  NO, KEEP LOOKING >>2586
  2586  ---
  2589  GET DEVICE COUNT (BF31)
  2580  IS BOOT ORIVE IN LIST? >>25A3
  258F  SO IT WILL BE SEARCHEO FIRST... (BF30)
  2592  STORE BOOT AT ENO OF SEARCH LIST (BF32)
  2596  ANY OTHERS? >>25AA
  2599  YES, SECOND DRIVE? >>25A3
  259D  STORE IT RIGHT BEHINO BOOT DRIVE (BF32)
  25A1  NOW ANY MORE? >>25AA
  25A3  ---
  25A4  YES, MOVE OTHERS AHEAO IN LIST (BF32)
  25AA  OO CHECKSUM ON ROM <2639>
  25AD  NOT AN AUTOSTART ROM? >>25B3
  25AF  AUTOSTART, STORE FINISHED MACHIO (BF98)
  25B2   ANO LEAVE
```

```
ProDOS Relocator -- V1.0.1 -- 1 JAN 84          NEXT OBJECT ADDR: 25B3
-------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------

  25B3  NONAUTOSTART, UNKNOWN MACHINE! >>23D4

25B6 *********** IDENTIFY I/D CARD ***********************************

  25B6  DO WE ALREADY RECOGNIZE THIS CARD? >>2618
  25B8  NO,
  25BA  CHECK SIGNATURE ON CARD FOR THUNDER CLOCK
  25BF  NOT IT? >>25DE
  25C5  THUNDER CLOCK, WHICH SLOT?
  25C9  SAVE SLOT NUMBER (LESS 1) (21FC)
  25CC  IN CLOCK CODE RELOCATION TABLES (2232)
  25D1  ENABLE CLOCK/CALENDAR JUMP IN GLOBALS (BF06)
  25D6  IS THERE A MACHID? >>25AA
  25D8  IF SO, MARK THAT A CLOCK IS PRESENT
  25DA  AND UPDATE MACHID
  25DC  GO MARK ROM IN THIS SLOT >>2618
  25DE  ---
  25E0  CHECK SIGNATURE OF MYSTERY CARD
  25E2  STANDARD BASIC SUPPORTED?
  25E4  NO, UNKNOWN CARD >>2607
  25E8  YES,
  25EA  DOUBLE CHECK BASIC SUPPORTED
  25EC  NO, UNKNOWN CARD >>2607
  25F0  YES,
  25F2  GENERIC SIGNATURE?
  25F4  NO, UNKNOWN CARD >>2607
  25F7  YES,
  25F9  80 COLUMN CARD?
  25FB  NO, UNKNOWN CARD >>2607
  25FF  GET MACHID IF WE HAVE ONE >>25AA
  2601  MARK 80 COLUMN CARD PRESENT
  2603  AND UPDATE MACHID
  2605  GO MARK ROM ON CARD PRESENT >>2618
  2607  UNKNOWN CARD, CHECK ROM TO...
  260B  SEE IF IT WILL HOLD A VALUE...
  2611  FOR SOME TIME.
  2618  IF SO, WE HAVE A CARD IN SLOT
  261A  CONVERT SLOT NUMBER...
  261D  TO A BIT POSITION (2631)
  2620  AND OR INTO SLTBYT (BF99)
  2626   RETURN TO CALLER

2627 *********** DATA AREA ******************************************

  2627  DISK DEVICE DRIVER ENTRY POINT
  2628  (2 BYTE ADDRESS)
```

```
ProDOS Relocator -- V1.0.1 -- 1 JAN 84          NEXT OBJECT ADDR: 2628
-------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------

  2629  DEVICE SIGNATURE FOR:
  262B  +0,+2,+4,+6 = THUNDERCLOCK
  262D  +1,+3,+5,+7 = DISK
  262F  (+7 NOT CHECKED)

        CHECKSUM FOR AUTOSTART RDM
  2631  BIT POSITION TABLE FOR SLOTS

2639 *********** COMPUTE AUTOSTART ROM CHECKSUM **********************

  2639  ---
  263A  GET ZERD IN INDEX REGISTER (2631)
  263D  SUM $FB09 ("APPLE II") IN ROM
  2644  UPDATE CHECKSUM (2631)
  264B  DD 8 BYTES IN ALL (2634)
  2651  MOVE LENGTH TO HIGH NIBBLE
  2656  AND COMBINE WITH CHECKSUM (2631)
  2659  FUDGE FACTOR
  265B  SHOULD COME OUT ZERO >>2660
  265D  IT DID...RETURN WITH MACHID
  265F  RETURN
  2660  ELSE, RETURN WITH ZERO MACHID
  2662  RETURN


2663 *********** RELOCATIDN ROUTINE **********************************
           (X/Y REGS CONTAIN TABLE ADDR)

  2663  SAVE PASSED TABLE ADDRESS
  2667  ---
  2669  GET OPERATION CODE
  266B  VALID OPERATION? (4 DR LESS)
  266D  NO, ERROR >>26E1
  2671  $14/15 --> OUTPUT BLOCK
  267B  $16/17 --> LENGTH
  2684  NEGATIVE LENGTH? >>26E3
  2686   CHECK DPERATION CODE
  2687  ZERD BLOCK? >>26EC
  268A  ND, $12/13 = $18/19 --> INPUT BLOCK
  2694  $1A/1B --> END OF INPUT BLDCK
  26A1  COPY BLOCK ONLY? >>2710
  26A3  SAVE RELOCATION DPERATIDN CODE (283C)
  26A9  SAVE NUMBER OF RANGES TO CHECK (283D)
  26AD  ---
  26AE  CDPY START PAGES TO TABLE
  26B9  ---
  26BA  AND END PAGES
  26C5  ---
  26C6  AND FINALLY, RELOCATIDN FACTORS
  26CE  BUMP TO NEXT TABLE ENTRY <2716>
```

```
ProDOS Relocator -- V1.0.1 -- 1 JAN 84          NEXT OBJECT ADDR: 26D1
-------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------

  26D1  RESTORE OPERATION CODE (283C)
  26D6  RELOCATE INSTRUCTIONS? >>26E6

26D8 ********** 2/3 - RELOCATE ADDRESSES ***************************

  26D8  NO, RELOCATE ADDRESS <277A>
  26DB  COPY BLOCK <2723>
  26DE  AND CONTINUE IF ALL WENT WELL >>2667
  26E1   NORMAL EXIT
  26E2  RETURN
  26E3  JUMP TO ERROR EXIT >>27B0

26E6 ********** 4 - RELOCATE INSTRUCTIONS **************************

  26E6  RELOCATE INSTRUCTIONS <278C>
  26E9  AND THEN COPY BLOCK >>26DB

26EC ********** 0 - ZERO BLOCK ***************************************

  26EC  BUMP TABLE POINTER TO NEXT ENTRY <2716>
  26F1  GET NUMBER OF PAGES TO DO
  26F3  NO FULL PAGES? >>2701
  26F6  ZERO AN ENTIRE PAGE
  26FB  BUMP PAGE POINTER
  26FD  AND DECREMENT LENGTH
  2701  GET LENGTH OF PARTIAL LAST PAGE
  2703  NO PARTIAL PAGE? >>270D
  2706  ZERO PARTIAL PAGE TOO
  270D  DONE, GET NEXT TABLE ENTRY >>2667

2710 ********** 1 - COPY BLOCK ***************************************

  2710  BUMP TABLE POINTER <2716>
  2713  AND GO COPY BLOCK >>26DB

2716 ********** ADVANCE TABLE POINTER ********************************

  2716   ADD FINAL ENTRY INDEX..
  271A  TO TABLE ENTRY ADDRESS
  2722  RETURN

2723 ********** COPY BLOCK ******************************************

  2723  ---
  2727  INPTR < OUTPTR? >>2734
  2729  NO, GREATER? >>2757
  272B  MSB'S ARE EQUAL, CHECK LSB'S ALSO
  2733   EXIT IF EQUAL
  2734  INPTR < OUTPTR, COPY LAST PAGES FIRST
  2738  BUMP BOTH INPTR AND OUTPTR BY...
  273A  LENGTH-1 TO POINT AT LAST BYTE
```

```
ProDOS Relocator -- V1.0.1 -- 1 JAN 84          NEXT OBJECT ADDR: 2742
-------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------

  2742  START WITH SHORT LAST PAGE LENGTH
  2746  ---
  2747  COPY BYTES BACKWARDS THROUGH MEMORY
  274E  DROP ADDRESSES AND LENGTH BY 256
  2754  AND CONTINUE UNTIL FINISHED >>2746
  2756  RETURN

  2757  INPTR > OUTPTR, COPY PAGES FORWARD
  2759  HOW MANY FULL PAGES LEFT?
  275B  NONE? >>276C
  275D  COPY A FULL PAGE
  2764  AND BUMP ADDRESSES
  2768  DECREMENT LENGTH BY 256
  276A  AND DO ALL PAGES >>275D
  276C  GET LENGTH OF LAST PAGE
  276E  EVEN PAGE BOUNDARY? >>2779
  2770  NO, COPY SHORT LAST PAGE
  2779  RETURN

277A ********** ADDR/PAGE RELOCATE **********************************

  277A  GET TABLE ENTRY TYPE (283C)
  277E  GET PAGE TO RELOCATE
  2780  RELOCATE A SINGLE ADDRESS <27B8>
  2783  BUMP BY 1 OR 2 BYTES (283C)
  2786  ADVANCE POINTER <27D4>
  2789  AND CONTINUE UNTIL COMPLETE >>277A
  278B  RETURN

278C ********** INSTRUCTIONS RELOCATE ********************************

  278C  ---
  278E  GET 6502 OPCODE
  2790  COMPUTE INSTRUCTION LENGTH <27E7>
  2793  INVALID OPCODE? >>27A6
  2795  3 BYTE INSTRUCTIONS?
  2797  NO >>27A0
  2799  YES, 3 BYTE ADDRESS TO CORRECT
  279B  RELOCATE ADDRESS <27B8>
  279E  AND ADVANCE BY 3 BYTES
  27A0  NEXT INSTRUCTION <27D4>
  27A3  CONTINUE UNTIL FINISHED >>278C
  27A5  RETURN

        ******** INVALID OPCODE *****************

  27A6   POP THE STACK
  27A8  RETURN WITH POINTER TO BAD INSTRUC.
  27AC  DIE HORRIBLY
  27AF  RETURN
```

```
ProDOS Relocator -- V1.0.1 -- 1 JAN 84          NEXT OBJECT ADDR: 27AF
-------------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------------

27B0 *********** ERROR RETURN ***************************************

  27B0  RETURN WITH POINTER
  27B4  EXIT WITH ERROR CODE
  27B7  RETURN

27B8 *********** RELOCATE ABSOLUTE ADDRESS ***************************

  27B8  GET PAGE NUMBER TO CHECK
  27BA  GET NUMBER OF RANGES (LESS ONE) (283D)
  27BD  IS IT PRIOR TO START OF THIS RANGE? (283E)
  27C0  YES? >>27C9
  27C2  NO, IS IS AFTER END OF RANGE? (2846)
  27C5  NO? >>27CD
  27C9  ---
  27CA  CHECK EACH RANGE >>27BD
  27CC  RETURN

  27CD  ---
  27CE  ADD FUDGE FACTOR TO ADDRESS (284E)
  27D1  AND UPDATE IT
  27D3  RETURN

27D4 *********** BUMP POINTER TO NEXT ADDR ***************************

  27D4  ---
  27D5  ADD LENGTH TO POINTER
  27DC  CHECK TO SEE IF WE ARE DONE
  27E2  ---
  27E6  RETURN

27E7 *********** COMPUTE INSTRUCTION LENGTH **************************

  27E7   A-REG CONTAINS OPCODE
  27E8  ISOLATE LAST TWO BITS FOR LATER
  27ED  USE LAST 6 BITS AS TABLE INDEX
  27EF  GET BYTE WITH 4 LENGTHS IN IT (27FC)
  27F2  ---
  27F3  USING TOP TWO BITS AS INDEX... >>27F9
  27F5  SHIFT DOWN THE PROPER LENGTH
  27F9  AND ISOLATE IT IN A-REG
  27FB  RETURN


27FC *********** 6502 OP LENGTH TABLE *********************************
          EACH BYTE CONTAINS FOUR 2 BIT LENGTHS
```

```
ProDOS Relocator -- V1.0.1 -- 1 JAN 84          NEXT OBJECT ADDR: 27FC
-------------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------------

  27FC  ---

283C *********** RELOCATION DATA ***************************************

  283C  RELOCATION CODE (3,2,1)
  283D  NUMBER OF RANGES
  283E   START OF RANGE PAGES
  2846   END OF RANGE PAGES +1
  284E   ADDITIVE FACTORS
  2856   NOT USED
  285A   PAD TO NEXT PAGE BOUNDARY
  28CB

28FF *********** SET UP RAMDRIVE IN AUXMEM ***************************

  28FF  ---
  2902  COPY RAM DRIVE DEVICE DRIVER TO.. (2C00)
  2905  THE LANGUAGE CARD (FF00)
  290F  $3C/3D --> $2A00
  2912  $3E/3F --> $2BFF
  291D  $42/43 --> $200
  2923   MAIN MEM TO AUX MEM COPY
  2924  COPY $2A00,L$200 TO AUXMEM $200 <C311>
  2929  SLOT 3, DRIVE 2 DEVICE DRIVER.. (BF26)
  292C  IS AT $FF00
  2931  BUMP DEVICE COUNT (BF31)
  2937  ADD DEVICE TO VOLUME SEARCH TABLE
  293C  RETURN

293D ********** 293D-29FF NOT USED ***********************************

  293D   NOT USED
  2969


2A00 *********** RAMDRIVE DEVICE DRIVER *******************************
          (COPIED TO $200 IN AUXILLARY MEMORY)

  2A00  SAVE THE 80STORE SETTING (C018)
  2A04  FORCE RAM READ/WRITE (C000)
  2A0C  COPY INPUT PARAMETERS
  2A14  FIRST TIME IN OR FORMAT COMMAND? (03BA)
  2A17  NO? >>2A4D

          ******** FORMAT RAMDRIVE ****************
```

```
ProOOS Relocator -- V1.0.1 -- 1 JAN 84          NEXT OBJECT AOOR: 2A19
------------------------------------------------------------------------
AOOR    OESCRIPTION/CONTENTS
------------------------------------------------------------------------

  2A19  YES, SAVE BLOCK WANTEO
  2A1O  ZERO SECTORS $E ANO $F <0331>
  2A22  COPY VOLUME NAME ($F3,"RAM") (0300)
  2A25  TO VOLUME OIRECTORY BLOCK (0E04)
  2A2B   $FF
  2A2E  HEX $FF'S TO TABLE (03C0)
  2A34  ZERO (03C0)
  2A39  COPY ACCESS/TOTAL BLOCKS TO (0304)
  2A3C  VOLUME OIRECTORY BLOCK (0E22)
  2A42  REFORMAT? (03BA)
  2A45  YES >>2AA8
  2A47  NO, OONE FIRST TIME PROCESSING (03BA)
  2A4A  RESTORE BLOCK NUMBER (03BF)

          ******** REAO/WRITE RAMORIVE BLOCK ******

  2A4O  BLOCK NUMBER * 2 = SECTOR NUMBER (03BF)
  2A53  SECTOR BEYONO MAIN MEMORY?
  2A55  YES >>2A61
  2A57  NO, SECTOR 6? (VOLUME BIT MAP)
  2A59  NO >>2A5E
  2A5B  YES, OUMMY UP A PHONEY BIT MAP SECTOR >>038A
  2A5E  ELSE, REAO/WRITE MAIN MEMORY SECTOR >>0340

          ******** REAO/WRITE IN LANG. CARO *******

  2A61   SAVE SECTOR NUMBER
  2A62  FINO IT IN MEMORY <02E3>
  2A65   REMEMBER REAO/WRITE STATUS
  2A66  WRITING? >>2AB6
0268
  2A68  ---
  2A69  NO, SECTOR FOLLOWS I/O 4K AREA?
  2A6B  YES >>2A71
  2A6O  NO, FORCE IT TO $OXXX
  2A6F  ANO USE 2NO BANK OF CARO >>2A77
  2A71  ELSE, USE 1ST BANK OF CARO (C083)
  2A74  ANO WRITE ENABLE IT (C083)
  2A77  SAVE SECTOR NUMBER IN BLOCK (03BF)
  2A7A  PRESERVE HIS BUFFER AOOR (03BE)
  2A7E  ACROSS THE FOLLOWING: (03BO)
  2A81  SELECT ALTERNATE ZEROPAGE (C009)
  2A86  USE $C00 AS A CROSS BANK XFER AREA (03BE)
  2A8B  PRETENO THAT WAS CALLER'S BUFFER (03BO)
  2A8E  ANO SET UP POINTERS AGAIN <02E3>
  2A92  COPY SECTOR TO $C00
  2A9O  THEN BACK TO MAIN ZERO PAGE (C008)
  2AA0  RESTORE CALLER'S BUFFER AOORESS (03BO)
  2AA7   REAOING OR WRITING?
  2AA8  IF WRITING, OONE >>2AB3
```

```
ProOOS Relocator -- V1.0.1 -- 1 JAN 84          NEXT OBJECT AOOR: 2AAA
------------------------------------------------------------------------
AOOR    OESCRIPTION/CONTENTS
------------------------------------------------------------------------

  2AAA  IF REAOING, ENABLE L.C. 1ST BANK (C08B)
  2AB0  ANO COPY BLOCK $C00 TO HIS BUFFER <02BC>
  2AB3  THEN EXIT >>030C
  2AB6  IF WRITING, COPY HIS BLOCK TO $C00 <02BC>
  2AB9  THEN OO COMMON COOE ABOVE >>0268

2ABC *********** COPY BLOCK IN MAIN 48K ******************************

02BC
  2ABC  THIS ENTRY COPIES SECTOR $0C/$O
02BE
  2ABE  THIS ENTRY COPIES ANY BLOCK (03BF)
02C1
  2AC1  FINO SECTOR/SET POINTERS <02E3>
  2AC4  WRITING? >>2A09
  2AC6  NO, WRITE TO MAIN 48K RAM (C004)
  2ACA  COPY BLOCK AUX MEM --> MAIN MEM
  2A05  WRITE TO AUX MEM AGAIN (C005)
  2A08   OONE
  2A09  ---
  2AOB  GO BACK TO MAIN MEM LANG. CARO.. (03EO)
  2AOE  TO COPY MAIN MEM --> AUX MEM

2AE3 *********** FINO RAM SECTOR/SET POINTERS ************************

02E3
  2AE3  GET COMMANO (03BB)
  2AE6  REAO OR WRITE?
  2AE7  WRITE? >>2B06
  2AE9  NO, REAO OR FORMAT (03BE)
  2AF0  $42/43 --> BUFFER IN HIS MEMORY (03BO)
  2AF3  $40/41 --> SECONO PAGE OF SAME
  2AF7  GET PAGE NUMBER (03BF)
  2AFC  $3C/3O --> BLOCK IN /RAM ORIVE
  2AFE  $3E/3F --> SECONO PAGE OF SAME
  2B04  ALWAYS BRANCH AROUNO WRITE COOE >>2B21

  2B06  WRITE,  (03BE)
  2B0O  $3C/3O --> BUFFER IN HIS MEMORY (03BO)
  2B10  $3E/3F --> SECONO PAGE OF SAME
  2B17  $42/43 --> BLOCK IN /RAM ORIVE
  2B19  $40/41 --> SECONO PAGE OF SAME
  2B21  SECONO PAGE FOLLOWS FIRST
  2B25   EXIT

2B26 *********** RETURN WITH OUMMY SECTOR ****************************
```

```
ProDOS Relocator -- V1.0.1 -- 1 JAN 84          NEXT OBJECT ADOR: 2B26
-------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------

  2B26  ZERO SECTOR $C/D AND SELECT IT <032F>
  2B29  COPY TO/FROM HIS BUFFER <02C1>
  2B2C  AND EXIT >>03DC

2B2F ********** ZERO BLOCK BUFFER **********************************

032F
  2B2F  ZERO SECTOR $C/D ENTRY
0331
  2B31  ZERO ANY GIVEN SECTOR ENTRY (03BF)
0334
  2B34  FIND SECTOR/SET POINTERS <02E3>
  2B38  ZERO BOTH PAGES OF BLOCK
  2B3F   AND EXIT

2B40 ********** READ/WRITE IN LOW 48K ******************************

0340
  2B40  SECTOR 4 (VOLUME DIRECTORY)?
  2B42  NO >>2B48
  2B44  YES, MAKE THAT BLOCK 7 INSTEAD
  2B46  AND GO DO I/O NOW >>2B56
  2B48  ELSE, LESS THAN SECTOR $0?
  2B4A  IF SO, PASS BACK $C ZEROED >>2B26
  2B4C  START MSB AT ZERO
  2B4E  GET ORIGINAL BLOCK NUMBER
  2B50  BLOCK $5D THROUGH $5F?
  2B52  NO >>2B59
  2B54  YES, ADJUST TO $0 THROUGH $F
  2B56  AND USE $1A00 THRU $1FFF IN /RAM >>0383
  2B59   ELSE, FOR SECTORS $0 THRU $5C
  2B5A  SUBTRACT 8
  2B5C  AND DIVIDE BY 17 ($11)
  2B62   XREG IS QUOTIENT
  2B66   AND AREG IS REMAINDER
  2B67  REMAINDER OF 1?
  2B69  NO >>2B71
  2B6B   YES, EVERY 17TH BLOCK GOES...
  2B6C   AT $1200, $1400, $1600, $1800
  2B6D  BY ADDING 8
  2B6F  AND GO DO IT >>2B83
  2B71   BUMP QUOTIENT (START AT $2XXX)
  2B73  SHIFT IT TO TOP NIBBLE OF BYTE
  2B7B  GOT A REMAINDER? >>2B7F
  2B7D   IF SO, DECREMENT IT (NOT USING 1)
  2B7F   THEN ADD INTO TOP NIBBLE
  2B80  TO FORM $14 THRU $4F (03BF)
0383
  2B83  BLOCK*2 FOR SECTOR NUMBER
  2B84  COPY THE BLOCK <02BE>
```

```
ProDOS Relocator -- V1.0.1 -- 1 JAN 84          NEXT OBJECT ADDR: 2B87
-------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------

  2B87  THEN EXIT >>030C

2B8A ********** READ/WRITE BIT MAP BLOCK ****************************

038A
  2B8A  USE $C/D AS A DUMMY SECTOR
  2B8F  GO FIND IT AND SET POINTERS <02E3>
  2B92  WRITING? >>2BA7
  2B94  NO, READING - ZERO BLOCK AT $C/D <0334>
  2B99  COPY BIT MAP IMAGE TO DUMMY BLOCK (03C0)
  2BA1  COPY BLOCK BACK TO CALLER'S BUFFER <02C1>
  2BA4  THEN EXIT >>03DC

  2BA7  WRITING, COPY CALLER'S BUFFER TO $C/D <02C1>
  2BAA  FIND $C/D AND SET POINTERS <02E3>
  2BAF  COPY FROM SECTOR TO BIT MAP IMAGE
  2BB7  THEN EXIT >>030C


2BBA ********** RAM DRIVE DATA (AT $3BA) ****************************

03BA
  2BBA  FIRST TIME ENTRY FLAG
03BB
  2BBB  COMMAND FROM PARM LIST
03BC
  2BBC  UNIT NUMBER FROM PARM LIST
03BD
03BE
  2BBD  BUFFER ADDRESS FROM PARM LIST
03BF
  2BBF  BLOCK NUMBER FROM PARM LIST

03C0
  2BC0  BIT MAP IMAGE FOR RAM DRIVE

03D0
  2BD0  /RAM VOLUME NAME
  2BD1  'RAM'
0304
  2BD4  ACCESS, ENTRY LENGTH
  2BD6  NUMBER OF ENTRIES
  2BD7  FILE COUNT
  2BD9  BIT MAP BLOCK POINTER
  2BDB  BLOCKS ON DISK
```

```
ProDOS Relocator -- V1.0.1 -- 1 JAN 84          NEXT OBJECT ADDR: 2BDC
-------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------

2BDC ********** EXIT TO MAIN MEMORY *********************************

03DC
  2BDC  WRITE ENABLE RAM CARD (C08B)
  2BE3  RESTORE 80STORE STATUS >>2BEA
  2BE5  80STORE WAS ON (C001)
  2BEA  GO AROUND PARM TO XFER >>03EF
03ED
  2BED  CROSS BANK XFER ADDRESS LSB
03EE
  2BEE  AND MSB
03EF
  2BEF  RETURN TO $FF44 (NORMAL EXIT)
03F6
  2BFB  USE ROM XFER ROUTINE TO DO IT >>C314

2C00 ********** DISK DEVICE DRIVER FOR /RAM *************************
         (COPIED TO $FF00 IN KERNEL)

  2C00  ---
  2C03  SAVE ZPAGE STUFF I WILL CLOBBER
  2C05  FROM $3C THRU $47 (FF81)
  2C0D  SAVE $3ED/E (CROSS BANK XFER ADDR) (03ED)
  2C16  COMMAND = STATUS?
  2C18  IF SO, SIMPLE EXIT WILL DO >>2C44
  2C1A  ELSE, TOO BIG A COMMAND NUM?
  2C1C  IF SO, ERROR >>2C3B
  2C1E  ELSE, INVERT BITS OF CMD
  2C20  AND SAVE IT
  2C22  FORMAT? >>2C2C
  2C24  NO, CHECK BLOCK NUMBER
  2C28  MUST BE <128 FOR /RAM
  2C2C  GOING TO $200 IN AUX MEMORY
FF33
  2C38  USE XFER TO GET THERE >>C314

  2C3B  I/O ERROR RETURN CODE
  2C3D  EXIT >>2C41
  2C3F  WRITE PROTECTED RETURN CODE
  2C41  ---
  2C42  ERROR EXIT >>2C47
  2C44  NORMAL EXIT, RETURN CODE IS 0
  2C47  ---
  2C4B  RESTORE ZERO PAGE IS USED (FF81)
  2C53  AND $3ED/E (FF7F)
  2C61   AND EXIT TO CALLER WHEN THRU
```

```
ProDOS Relocator -- V1.0.1 -- 1 JAN 84          NEXT OBJECT ADDR: 2C61
-------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------

2C62 ********** COPY MAIN TO AUX BLOCK *******************************
          (CALLED FROM AUX MEM HANDLER)

FF62
  2C62  WRITE IN AUX 48K (C005)
  2C67  COPY BOTH PAGES OF BLOCK
  2C72  WRITE IN MAIN 48K AGAIN (C004)
  2C77  GO TO $2D8 IN AUX MEMORY TO RETURN (03ED)
  2C7C  RETURN TO AUX MEM HANDLER AGAIN >>FF33

2C7F ********** DATA AREA ********************************************

FF7F
  2C7F  SAVED XFER ADDRESS
FF80

FF81
  2C81   ZERO PAGE SAVE AREA

2C8D ********** NOT USED *********************************************

  2C8D  ---

2D00 ********** START OF PRODOS LOAD IMAGE **************************

2D00    LOAD IMAGE AT $2D00

  2D00  ---
```

```
ProDOS MLI -- V1.0.1 -- 1 JAN 84                NEXT OBJECT ADDR: D000
-------------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------------

D000    MODULE STARTING ADDRESS

          ****************************************
          *                                      *
          *  PRODOS MACHINE LANGUAGE INTERFACE   *
          *       THIS CODE NORMALLY RESIDES IN  *
          *       THE LANGUAGE CARD (FOR 64K)    *
          *       IT PERFORMS ALL FILE MANAGEMENT*
          *       AND OTHER SYSTEM FUNCTIONS AND *
          *       SUPPORTS THE HARDWARE IN A     *
          *       DEVICE INDEPENDENT WAY.        *
          *                                      *
          *  VERSION 1.0.1 -- 1 JAN 84           *
          *                                      *
          ****************************************

D000 ********** ZERO PAGE USEAGE ************************************

0040    Pointer to callers parmlist
0041
          -- device driver parmlist --
0042    Command
0043    Unit Number
0044    Buffer Pointer
0045
0046    Block Number
0047
          ----------------------------
0048    I/O Pointer - Index Block or..
0048    pointer into $F600 work buffer or..
0048    caller's pathname buffer pointer
0049
004A    I/O Pointer - Data Block
004B
004C    I/O Pointer - Data Block
004D
004E    I/O Pointer - Caller's Data or..
004E    buffer pointer passed in parmlist or..
004E    old I/O buffer
004F

D000 ********** MLI ERROR CODES **************************************

0000    No Error
0001    Bad call type
0004    Bad parameter count
0025    Interrupt Table full
0027    I/O Error
0028    No device connected
002B    Write protected
002E    Volume switched
```

```
ProDOS MLI -- V1.0.1 -- 1 JAN 84                NEXT OBJECT ADDR: D000
-------------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------------

0040    Invalid pathname syntax
0042    Too many files open
0043    Invalid REF NUM
0044    Nonexistent path
0045    Volume not mounted
0046    File not found
0047    Duplicate file name
0048    Disk full
0049    Volume Directory full
004A    Incompatible ProDOS version
004B    Unsupported file type
004C    End of file
004D    Position past EOF
004E    Access error
0050    File already open
0051    File count bad
0052    Not a ProDOS disk
0053    Bad parameter
0055    VCB overflow
0056    Bad buffer address
0057    Duplicate volume mounted
005A    Bad volume bit map

D000 ********** SCREEN LOCATIONS *************************************

0750    For direct movement of text to screen
07D0
07F1
07F2
07F3
07F4
07F5
07F6
07F7
07F8    Slot in use

D000 ********** SYSTEM GLOBAL PAGE EQUATES ***************************

BF00    Jump to MLI entry point
BF03    JSPARE (Jump to $EECF, QUIT code)
BF06    DATETIME vector
BF09    Jump to System Error
BF0C    Jump to System Death Handler
BF0F    System Error number
BF10    Device Driver address table
BF30    Slot/Drive last device
BF31    Count (-1) active devices
BF32    List of active devices by DEVID
BF58    Memory BITMAP for low 48K
BF70    Open file 1 buffer address
BF7E    Open file 8 buffer address
```

```
ProDOS MLI -- V1.0.1 -- 1 JAN 84                NEXT OBJECT ADDR: D000
-------------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------------

BF80    Interrupt handler 1
BF82    Interrupt handler 2
BF84    Interrupt handler 3
BF86    Interrupt handler 4
BF88    A reg save during interrupt
BF89    X reg save during interrupt
BF8A    Y reg save during interrupt
BF8B    S reg save during interrupt
BF8C    P reg save during interrupt
BF8E    Interrupt return address
BF90    Date/Time
BF94    File open LEVEL
BF95    Backup bit
BF9A    Prefix flag (0 = no prefix)
BF9B    MLI active flag
BF9C    Last MLI call return address
BF9E    MLI X reg savearea
BF9F    MLI Y reg savearea
BFA0    Language card entry/exit routines
BFD0    Interrupt entry/exit routines
BFF4    Bank switch saved state ($E000 byte)
BFFF    Kernel version number

D000 ********** SOFT SWITCHES ****************************************

C00C    Reset 80 column mode
C051    Set TEXT mode
C053    Set Mixed text/graphics
C054    Display Primary page
C056    Set LORES graphics mode
CFFF    Reset alternate I/O ROMs

D000 ********** MLI MAIN ENTRY POINT *********************************

  D000   Clear decimal mode
  D001  Save Registers (BF9F)
  D007   Set ($40) -> Address of function code -1
  D00B  Set CMDADR -> True return address
  D01A  Init Global Page System error to 0 (BF0F)
  D01E  Get Function Code
  D021  Build hash index into Command Table (X reg)
  D02A  Is this code valid?
  D02F  No >>D0A7
  D032  Set ($40) -> Parameter list
  D03F  Get parameter count required (EF45)
  D042  None? >>D060
  D044  No - is parameter count correct?
  D046  No >>D0AB
  D048  Check class of function (EF25)
  D04B  Quit?
  D04D  yes >>D05D
```

```
ProDOS MLI -- V1.0.1 -- 1 JAN 84                NEXT OBJECT ADDR: D04F
-------------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------------

  D04F  no,
  D050  $8X - Calls to I/O Drivers >>D066
  D052  $CX/DX - Non System calls >>D071
  D054  Else, $4X - Interrupt support
  D055  Isolate type (DEALLOC = 1, ALLOC = 0)
  D057  Call Interrupt Support <D0F3>
  D05A  Then Exit to Caller >>D078
  D05D  Go to quit code via global page >>BF03

D060 *************************************************************
         ****** MLI GET_TIME CALL ******
         ******************************

  D060  Call Date/Time driver <BF06>
  D063  and exit to caller >>D078

D066 *************************************************************
         ****** MLI READ_BLOCK CALL *******
         ****** MLI WRITE_BLOCK CALL ******
         ********************************
         $80 - Read Block
         $81 - Write Block

  D066  ---
  D067  Set $42 -> 1 for READ, 2 for WRITE
  D06B  Do Block I/O <D0B2>
  D06E  Then Exit to Caller >>D078

D071 ********** $CX and $DX CALLS ************************************

  D071  ---
  D072  Isolate function Index
  D075  Perform function and exit to caller <D23C>

D078 ********** EXIT TO CALLER ***************************************

  D078  Clear Backup
  D080  Error occurred?
  D083   Save test results
  D084   Disable interrupts
  D085  MLI no longer active (BF9B)
  D088   Get test results back
  D089   Store in X reg
  D08A  Set up Return Address on stack (BF9D)
  D092   Put test results on stack
  D094   Put error code in A reg
  D095  Restore X reg (BF9E)
  D098  Restore Y reg (BF9F)
  D09B   Put error code on stack
  D09C  Get RAM/ROM orientation (BFF4)
  D09F  Exit via RAM Global Page >>BFA0
```

```
ProDOS MLI -- V1.0.1 -- 1 JAN 84                NEXT OBJECT ADDR: D09F
-------------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------------

D0A2 ********** ND DEVICE CONNECTED **********************************

  D0A2  ---
  D0A4  Call System Error Handler (Global Page) <BF09>

D0A7 ********** BAD SYSTEM CALL NUMBER ********************************

  D0A7  ---
  D0A9  Branch always taken >>D0AD

D0AB ********** BAD PARAMETER COUNT **********************************

  D0AB  ---
  D0AD  Call System Error Handler <D0D7>
  D0B0  Exit to Caller >>D078

D0B2 ********** BLOCK I/O SETUP ***************************************

  D0B2  ---
  D0B4   Save Old Processor Flags
  D0B5   Disable Interrupts
  D0B6  Copy Parameters to $43-$47
  D0BE  Save Starting Buffer Page in $4F
  D0C3   Find last page + 1
  D0C6  Round up if Buffer not page aligned >>D0C9
  D0C9  Is this Memory already in use? <EE89>
  D0CC  Yes, then exit with error >>D0D6
  D0CE  No, do Block I/O <D0DA>
  D0D1  Error? >>D0D6
  D0D3   No, then exit normally
  D0D5  RETURN
  D0D6   Error Exit
  D0D7  Call System Error Handler <BF09>

D0DA ********** Block I/O *****************************************

  D0DA  ---
  D0DC  Force off unused UNIT bits
  D0E3   Put Drive number in X reg
  D0E7  Put Device Handler Address in Jump Vector (F0B5)
  D0F0  Exit through Device Handler >>F0B5

D0F3 ********** Interrupt Handler **************************************
          ALLOC/DEALLOC

  D0F3  Save Call Type
  D0F5  Which Type?
  D0F6  DEALLOC? >>D124
```

```
ProDOS MLI -- V1.0.1 -- 1 JAN 84                NEXT OBJECT ADDR: D0F6
-------------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------------

          ALLOC

  D0F8  ---
  D0FA  Look for empty slot (BF7E)
  D101  His Address better be non-zero
  D105  Store Address of His routine in Global Page (BF7E)
  D10E   And return the position number we used
  D114   Exit
  D115   Skip this Vector
  D117  Last one?
  D119  No, check another >>D0FA
  D11B  Yes, Table Full Error
  D11D  Always taken >>D121
  D11F  Bad Parameter Error
  D121  Call System Error Handler <BF09>

          DEALLOC

  D124  ---
  D126  Get Position Number
  D128  Can't be zero >>D11F
  D12C  Or greater than 4 >>D11F
  D12F   Make Index into Table from it
  D132  And zero His Vector (BF7E)
  D139   Then Exit

D13A ********** IRQ Handler ****************************************

  D13A  ---
  D13C  Save A reg from Monitor (BF88)
  D13F  And X,Y,S and P (BF89)
  D14E  And RTI Address (BF8E)
  D155   Replace stack to original condition
  D159  Save active slot index (D1C4)
  D15C   Is stack full?
  D15F  Pop off 16 bytes and save them
  D161  ---
  D168  Save $FA - $FF (top of zero page)
  D16A  ---
  D172  Is there a User Vector #1 (BF81)
  D175  No >>D17C
  D177  Yes, call it <D1CE>
  D17A  His interrupt? >>D19F
  D17C  Is there a User Vector #2 (BF83)
  D17F  No >>D186
  D181  Yes, call it <D1D1>
  D184  His interrupt? >>D19F
  D186  Is there a User Vector #3 (BF85)
  D189  No >>D190
  D18B  Yes, call it <D1D4>
```

```
ProDOS MLI -- V1.0.1 -- 1 JAN 84                NEXT OBJECT ADDR: D18E
-------------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------------

  D18E  His interrupt? >>D19F
  D190  Is there a User Vector #4 (BF87)
  D193  No >>D19A
  D195  Yes, call it <D1D7>
  D198  His interrupt? >>D19F
  D19A  Indicate error type 1
  D19C  Call System Death Handler <BF00>
  D19F  Interrupt Serviced
  D1A1  Restore zero page (EFA5)
  D1A9  And stack (BF8B)
  D1B9  Reload X and Y (BF8A)
  D1BF  Disable I/O ROMS (CFFF).
  D1C2  Replace active slot number (C100)
  D1CB  Exit from Interrupt >>BFD0
  D1CE  User Interrupt Handlers (#1 - #4) >>BF80

D1DA ********** SYSTEM ERROR HANDLER **********************************

  D1DA  Save Error Code (BF0F)
  D1DE   Pop out of subroutine
  D1DF  Exit to caller with Error Code (BF0F)
  D1E3  RETURN

D1E4 ********** SYSTEM DEATH HANDLER **********************************

  D1E4  ---
  D1E6   ; Entry from System Global Page here
  D1E7  Turn off 80 column card (C00C)
  D1EA  Select standard Text display (C051)
  D1F6  Blank out a line
  D1F8  ---
  D1FD  Print "INSERT SYSTEM DISK AND RESTART" (EFDE)
  D207  Go into infinate loop if no error code >>D239
  D20B  "-" (07F1)
  D210  "E" (07F2)
  D215  "R" (07F3)
  D218  "R" (07F4)
  D21C  Convert error code to Hex
  D228  And print it (07F6)
  D22C  Second digit also
  D239  Infinite loop >>D239

D23C ********** PERFORM FILING OR ************************************
          ******* HOUSEKEEPING FUNCTIONS ******

  D23C  Save function index (F077)
  D23F  Get INFO flags for this command (EF8D)
  D242  Times 2
  D243  Store Command Number times 2 (F073)
  D248   And use it to index into Address Table
  D24C  Set up Jump Vector with this funtion's (F0B5)
```

```
ProDOS MLI -- V1.0.1 -- 1 JAN 84                NEXT OBJECT ADDR: D24F
-------------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------------

  D24F  ..handler address (EF66)
  D255  Signal Backup required after call
  D25A  PATHNAME not required? >>D261
  D25C  Required - parse and validity check <D27F>
  D25F  Bad Name? >>D278
  D261  Reference Number in list? (F073)
  D264  No >>D26B
  D266  Yes - check it out <D3C5>
  D269  Bad Number? >>D278
  D26B  Date/Time in list? (F073)
  D26E  No >>D273
  D270  Yes - set System date just in case <BF06>
  D273  Call Function Handler <D27C>
  D276  If no errors the exit >>D27B
  D278  Else - call System error handler <BF09>
  D27B   Return to caller
  D27C  Indirect JUMP to Handler >>F0B5

D27F ********* CHECK CALLER'S PATHNAME  ******************************
          ****** COPY TO MY AREA          ******

  D27F  Set ($48) -> Pathname
  D28A  ---
  D28E  Assume partial Pathname (F07C)
  D291  No Pathname in my area yet  (F100)
  D294  Check length of caller's Pathname
  D296  Zero is no good >>D2F0
  D29A  Nor is 65 or more >>D2F0
  D29C  Save length (F05E)
  D29F  Length + 1 (F05E)
  D2A3  Get first character of his name
  D2A7  Is it "/"?
  D2A9  No >>D2AF
  D2AB  Yes - indicate fully qualified name (F07C)
  D2AE   Bump past "/"
  D2AF  ---
  D2B1  Length of Index level is -1 initially (F100)
  D2B4  First character of Index level (counter) (F078)
  D2B7  Start of upcomming Index level in name (F07A)
  D2BA  At end of name yet? (F05E)
  D2BD  Yes >>D2F4
  D2BF  No - get next character in his name
  D2C5  Is it "/"?
  D2C7  Yes >>D309
  D2C9  No - lower case?
  D2CB  No >>D2CF
  D2CD  Yes - force upper case
  D2CF  Copy to my Pathname buffer (F100)
  D2D2  Increment Index level counter (F078)
  D2D5  Subsequent characters may be A-Z,0-9 or . >>D2DC
  D2D7  Increment Index level counter (F078)
```

```
ProDOS MLI -- V1.0.1 -- 1 JAN 84                NEXT OBJECT ADDR: D2DA
------------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
------------------------------------------------------------------------

  D2DA  First character must be alphabetic >>D2E8
  D2DC  Is it "."?
  D2DE  Yes - get next character >>D2BA
  D2E0  No - is it special or control character
  D2E2  Yes - Bad Pathname then >>D2F0
  D2E4  Is it numeric?
  D2E6  Yes - get next character >>D2BA
  D2E8  Is it Alphabetic?
  D2EE  If so get next character >>D2BA
  D2F0   Else
  D2F1  Bad Pathname
  D2F3  RETURN
  D2F4  ---
  D2F6  Any characters in last Index level? (F078)
  D2F9  Yes >>D2FF
  D2FB  No, zero characters in it (F078)
  D2FE   And toss out last "/"
  D2FF  ---
  D300  Mark end of name with $00 (F100)
  D303  Name too long? >>D2F0
  D305  No - save final length (F05E)
  D308   Set X -> 0
  D30C  Last Index more than 15 characters?
  D30E  Yes - then no good >>D2F0
  D310  Save output Index (F07D)
  D313  Store length of previous Index level (F07A)
  D316  Just before it in buffer (F100)
  D319  Restore output index (F07D)
  D31C  And continue >>D2AF
  D31E   End of Name
  D31F  Fully qualified name? (F07C)
  D322  Yes >>D329
  D324  No - Got a Prefix (BF9A)
  D327  No - error >>D2F0
  D329   Else, okay to exit

D32A ************************************************************
       ****** MLI SET_PREFIX CALL ******
       ********************************

  D32A  Copy Pathname <D27F>
  D32D  Its okay >>D339
  D32F  Check length of Volume name (F100)
  D334  If zero - no Prefix wanted (BF9A)
  D337   Exit with no error
  D338  RETURN
```

```
ProDOS MLI -- V1.0.1 -- 1 JAN 84                NEXT OBJECT ADDR: D338
------------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
------------------------------------------------------------------------

  D339  Get File entry for last index <D798>
  D33C  Okay? >>D342
  D33E  Invalid Pathname?
  D340  No - Out now! >>D380
  D342  Sub Directory file? (F01F)
  D349  No, error >>D37E
  D34B  Fully qualified path? (F07C)
  D34E  Yes >>D353
  D350  No - use old Prefix also (BF9A)
  D353  ---
  D355  Compute new Prefix Index (F05E)
  D358  Does new Prefix exceed 64 characters?
  D35A  Yes - Bad Path error >>D2F0
  D35D  Store new Prefix pointer (BF9A)
  D363  Set Device Number of Prefix Directory (F05F)
  D369  Save Keyblock for Prefix Directory (F060)
  D372  Copy Prefix to top of Path buffer (F100)
  D375  (preceeded by old Prefix if one exists) (F100)
  D37D   Exit normally

  D37E  Bad File Type Error
  D380  ---
  D381  RETURN

D382 ************************************************************
       ****** MLI GET_PREFIX CALL ******
       ********************************

  D382   Set ($4E) -> Data Buffer
  D38E  Set Length = 64 (max)
  D398  Validity check buffer storage <EE6C>
  D39B  Error? >>D380
  D39F  Get Prefix index (BF9A)
  D3A3  No Prefix? - Length = 0 >>D3A9
  D3A5  Compliment for length
  D3A9  Store in first byte of buffer
  D3AB  If null Prefix exit >>D3C3
  D3AD  ---
  D3AE  Copy Prefix to caller's buffer replacing (F100)
  D3B1  index level name length bytes with "/"
  D3BB  ---
  D3BF  End it with a "/"
  D3C3  ---
  D3C4   Exit normally
```

```
ProDOS MLI -- V1.0.1 -- 1 JAN 84                NEXT OBJECT ADDR: D3C5
-------------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------------

D3C5 ********** VALIDITY CHECK REFERENCE NUMBER **********************
      ,             (PASSED BY CALLER)

  D3C5  Get Reference Number
  D3C9  If zero then no good >>D426
  D3CD  If > 8 then no good >>D426
  D3CF   Save Reference Number
  D3D0  Multiply by 32
  D3D6  Result gives offset into FCB's (F052)
  D3DA   Get back Reference Number
  D3DB  File Control Block active this Reference? (F300)
  D3DE  No - Bad Reference Number  >>D421
  D3E0  Get Buffer Number (F30B)
  D3E3  Find Buffer address in Global Page <EE26>
  D3E9  No Buffer? >>D412
  D3EB  Buffer okay, save Page Pointer in $48
  D3EF  Second block in $49
  D3F1  Set last Device used in Global Page (F301)
  D3F7  Finish setting up pointers (F09D)
  D3FA  ($4A) -> 1st Block of Buffer (data)
  D3FC  ($48) -> 2nd Block of Buffer (index)
  D3FE  ---
  D3FF  Search all Volume Control Blocks (F210)
  D402  for the one which goes with requested unit (F301)
  D407  ---
  D40D  Can't find matching Volume Control Block
  D40F  So die with error type $0A <BF0C>
  D412  No Buffer in open File Control Block
  D414  So die with error type $0B <BF0C>
  D417  Is Volume mounted? (F200)
  D41A  No, keep looking >>D407
  D41C  Save Volume Control Block index (F051)
  D420   Exit normally

  D421  ---
  D423  !!!!! (F052)
  D426  Bad Reference Number error
  D429  RETURN

D42A ************************************************************
        ****** MLI ONLINE CALL ******
        ****************************

  D42A  Set ($4E) -> Data Buffer <E403>
  D42D  Set Length = 0
  D437  Get Unit Number
  D439  Do all Units? >>D442
  D43B  No, just one
  D43D  Set length = 16 (F09A)
  D440  Always taken >>D447
  D442  If all Units
```

```
ProDOS MLI -- V1.0.1 -- 1 JAN 84                NEXT OBJECT ADDR: D444
-------------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------------

  D444  Set Length = 256 (maximum) (F09B)
  D447  Is Buffer in main RAM? <EE6C>
  D44A  No, then exit >>D47F
  D44C  Yes, zero out Buffer
  D451  ---
  D456  Index into Data Buffer = $00 (F07A)
  D45B  Get Unit Number again
  D45D  Isolate valid bits
  D45F  Specific Unit requested? >>D480
  D461  No, copy Devise List from Global Page <DA57>
  D464  Save Device Count (F07D)
  D467  Get last Device (F08A)
  D46A  Generate return data for it <D480>
  D46D  Bump data buffer index by 16 (F07A)
  D476  Get next Device (F07D)
  D47A  And go do it >>D464
  D47C  When done, exit
  D47F  RETURN

  D480  Save Device Number (BF30)
  D483  Scan for the Volume Control Block <DA69>
  D486  Error? >>D4B8
  D488  We need Block 2 (Key Block of VolDir)
  D490  Read Volume Directory Key Block <DDE1>
  D493  Error? >>D4B8
  D495  Was something already mounted? (F051)
  D49B  No >>D4A2
  D49D  Yes, Files open? (F211)
  D4A0  Yes >>D4AE
  D4A2  No, set up Volume Control Block for new VOL <DAC4>
  D4A5  Error? >>D4B8
  D4A7  No
  D4A9  Was a duplicate Volume Control Block found? (F075)
  D4AC  Yes, then error >>D4B8
  D4AE  See if the same Volume is still there (F051)
  D4B4  If not, Disk Switch Error
  D4B6  Else, all is well - continue >>D4D6

D4B8 ********** ERROR ************************************************
        Store code in data buffer entry

  D4B8  ---
  D4B9  Store Device Number in entry <D4EB>
  D4BE  Store error code next
  D4C0  Duplicate Volume error?
  D4C2  No - done >>D4D4
  D4C5  Store Device Number for duplicate next (F076)
  D4CD  No Duplicate now
  D4D4   Exit with error
  D4D5  RETURN
```

```
ProDOS MLI -- V1.0.1 -- 1 JAN 84                NEXT OBJECT ADDR: D4D5
-------------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------------

D4D6 ********** MAKE ONLINE VOLUME ENTRY ****************************

  D4D6  Get name length for loop index (F200)
  D4DF  Copy name to Buffer entry (F200)
  D4E6  Done yet? (F078)
  D4E9  No, do another >>D4DF
  D4EB  Yes, find current Buffer entry (F07A)
  D4EE  Store Device number (BF30)
  D4F6   Return to caller

D4F7 ************************************************************************
          ****** MLI CREATE CALL ******
          ****************************

  D4F7  Follow Path to File <D7AB>
  D4FA  Error? - I'm expecting one >>D500
  D4FC  If File was found - Duplicate error
  D4FE  ---
  D4FF   Return to caller

  D500  File not found?
  D502  No, then a real error occured >>D4FE
  D504  Yes, get requested storage type
  D508  Is it 00, $01, $02 or $03?
  D50A  Yes, carry on >>D510
  D50C  Is it $0D?
  D50E  No, then exit with error >>D520
  D510  Get status of this device (BF30)
  D516  Exit on error >>D523
  D518  Is there a free Directory entry? (F05B)
  D51B  No >>D524
  D51D  Yes - continue >>D5B6

  D520  Indicate Bad Storage Type
  D523   Return to caller

  D524  Is this the Volume Directory? (F006)
  D52A  No, we can extend it >>D530
  D52C  Yes, indicate Volume Directory Full error
  D52F   Return to caller

          * EXTEND DIRECTORY FILE *

  D530  Save old current Block number
  D536  Allocate a Block on Disk <DCA9>
  D539   Save the number
  D53A   Replace BLKNUM
  D540   Was there a free Block?
  D541  No, then exit >>D523
  D543  Yes, set up forward pointer in old one (F602)
```

```
ProDOS MLI -- V1.0.1 -- 1 JAN 84                NEXT OBJECT ADDR: D546
-------------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------------

  D546  to point to it (F603)
  D549  and Write old Directory Block <DDDD>
  D54C  Error? Yes, then exit >>D523
  D550  Set BLKNUM -> new Block number
  D555  Back point to old Directory Block (F602)
  D55B  Loop until done >>D550
  D55F  Zero remainder of Block Buffer (F602)
  D562  (including forward pointer) (F700)
  D566  Loop until done >>D55F
  D568  Write new Directory Block <DDDD>
  D56B  Error? Yes, then exit >>D523
  D56D  Set BLKNUM -> Parent Directory number (F006)
  D577  Read Block with my entry <DDE1>
  D57A  Entry number of my Directory (F008)
  D57D  None relocatable!!
  D57F  Set ($48) -> Buffer
  D581  Skip link pointers
  D583  ---
  D584   Count entries
  D587  Skip to next (F009)
  D590  Save LSB
  D594  Add 1 to Blocks used
  D596  and $200 to EOF mark (EF8E)
  D599  in entry
  D59F  Loop until done >>D594
  D5A1  Write back Block to Parent Directory <DDDD>
  D5A4  Error? then exit  >>D5B5
  D5A6  Start all over now that there's room >>D4F7

D5A9 ********** ZERO $F600 ***************************************

  D5A9  Zero $F600 Block Buffer
  D5B5   Return to caller

D5B6 ********** BUILD NEW FILE ***********************************

  D5B6  Call Zero $F600 routine <D5A9>
  D5B9  Copy Datetime (Creation)
  D5BB  to my variables
  D5C7  Loop until done >>D5BB
  D5C9   Did he give Datetime (Creation)?
  D5CA  Yes, carry on >>D5D7
  D5CC  No, then use
  D5CE  System Datetime instead (BF90)
  D5D7  If Storage type is $00, $01, $02 or $03
  D5D9  force it to $10
  D5DF  else use a $D0
  D5E1  Find File name (F07A)
  D5E4  OR Storage type to name length (F100)
  D5E7  Store Type/Length (F01F)
  D5EA  Isolate name length
```

```
ProDOS MLI -- V1.0.1 -- 1 JAN 84                NEXT OBJECT ADDR: D5EE
-------------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------------

  D5EE  Copy File name to File Entry Buffer (F07A)
  D5FC  Copy caller's Access Byte
          NOTE: This should be validity checked!!!
  D604  and copy File type
  D609  ---
  D60A  and AUX_TYPE
  D613  Copy Version and Min_Version (0,0) (EFB0)
  D616  constants to entry (F03B)
  D61F  Indicate 1 Block used
  D624  Copy Directory Header Block number (F01A)
  D633  Is this a Seedling file?
  D635  Yes >>D66E
  D637  No, Directory file - Build Header in $F600
  D639  Copy completed Directory entry (F01F)
  D63C  to $F600 buffer first (F604)
  D640  Loop until done >>D639
  D642  Make Storage type $E in Header itself
  D647  Put "HUSTON" (Author) in Reserved area
  D64F  and Version, Min_Version, Access, (EFB0)
  D652  Entry-length, File count and (F620)
  D655   Parent pointer from constants
  D656  Loop until done >>D649
  D65A  Copy Parent Block entry number (F01C)
  D661  Loop until done >>D65A
  D663  Copy Parent entry Length (F011)
  D66B  EOF = $200 (F035)
  D66E  Allocate a new disk block <DCA9>
  D671  error? >>D6AA
  D673  Store it in key pointer of entry (F030)
  D679  and in BLKNUM for I/O
  D67D  Write zeroed (or DIR HDR) key block <DDDD>
  D680  error? >>D6AA
  D682  Bump parent's file count (F013)
  D68A  Go update directory <D6AB>
  D68D  error? >>D6AA
  D68F  Checkpoint Volume Bit Map and exit >>DD86

D692 ********** POINT $48/49 AT DIRECTORY ENTRY **********************

  D692  $48/$49 --> Entry
  D696  Skip link pointers (+4)
  D698  File entry number counter (F01E)
  D69B  ---
  D69C   Skip to proper entry
  D69F  Add entry length (F011)
  D6A4  (bump MSB)
  D6A8  (store LSB)
  D6AA  RETURN
```

```
ProDOS MLI -- V1.0.1 -- 1 JAN 84                NEXT OBJECT ADDR: D6AA
-------------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------------

D6AB ********** UPDATE DIRECTORY(S) **********************************

  D6AB  System date avaiable? (BF90)
  D6AE  no, forget it >>D6BB
  D6B2  yes, copy to last modified date field (BF90)
  D6BB  turn on BUBIT (backup) if appropriate (F03D)
  D6C4  set DEVNUM of parent (F019)
  D6CA  and BLKNUM (F01C)
  D6D4  reread DIR block containing entry <DDE1>
  D6D7  error? >>D6AA
  D6D9  Point to proper entry in buffer <D692>
  D6E0  Copy constructed entry to buffer (F01F)
  D6EB  Is this block the DIR HDR block?
  D6F6  no, write back new entry <DDDD>
  D6F9  error? >>D6AA
  D705  and then read DIR HDR block <DDE1>
  D708  error? >>D6AA
  D70A  in any case..
  D70C  copy back update file count to HDR (F013)
  D715  and ACCESS byte (with Backup) (F010)
  D71B  write back HDR block <DDDD>
  D71E  error? >>D778
  D720  is this the VOL DIR? (F604)
  D727  yes, all done -- exit >>D796
  D729  no, subdirectory.. (F627)
  D72C  get parent pointer
  D733  get parent entry no.. (F629)
  D739  and entry len (F62A)
  D73F  read parent DIR block <DDE1>
  D742  error? >>D778
  D744  find entry for this subdirectory <D692>
  D747  system date available? (BF90)
  D74A  no >>D759
  D74C  yes,
  D750  copy system date/time to... (BF90)
  D753  modified date/time in entry
  D759  write it back <DDDD>
  D75C  error? >>D778
  D760  BLKNUM = HDR block number
  D769  same block we have now?
  D76D  yes, go back and date stamp >>D720
  D76F  no,
  D773  read HDR block <DDE1>
  D776  and go back to date stamp parent DIR >>D720
  D778   error? then exit
```

```
ProDOS MLI -- V1.0.1 -- 1 JAN 84                NEXT OBJECT ADDR: D778
-------------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------------

D779 ********** NOT ProDOS VOLUME ERROR ******************************

  D779  ---
  D77C  RETURN

D77D ********** IS THIS ProDOS VOLUME? *******************************

  D77D  Does previous block ptr = 0? (F600)
  D78B  no, not a ProDOS volume >>D779
  D78D  else, (F604)
  D792  does VOL DIR's STORAGE TYPE = $E or $F?
  D794  no, error >>D779
  D796   else, ok
  D797  RETURN

D798 ********** GET FILE ENTRY ***************************************

  D798  follow path to it's end <D7AB>
  D79B  error? >>D7AA
  D7A0  copy file entry
  D7A8  and exit
  D7AA  RETURN

D7AB ********** FOLLOW PATH TO A FILE ********************************

  D7AB  get base dir's data <D92F>
  D7AE  error? >>D802
  D7B0  another subdirectory in the path? >>D7DA
  D7B2  no, at end of path (D82A)
  D7B5  $48/$49 --> $F604 (HDR)
  D7BD  copy part of HDR to file entry
  D7C7  File type = $F (Directory) (EFA8)
  D7CA  BLOCK = 2 (F01F)
  D7CD   No. blocks used = 4
  D7CE  EOF = $800
  D7D2  TYPE = subdirectory ($D0)
  D7D7  return to caller
  D7D9  RETURN

          *** SCAN DIRECTORY FOR FILE ***

  D7DA  indicate no free entry found as yet
  D7DF   signal in HDR block
  D7E0  zero count of names examined
  D7E5  find name in block <D8D8>
  D7E8  got it! >>D84F
  D7EA  not yet, how many entries expected? (F058)
  D7ED  less entry no. I just searched (F057)
  D7F2  more file entries left to search? >>D804
  D800  no, directory error
```

```
ProDOS MLI -- V1.0.1 -- 1 JAN 84                NEXT OBJECT ADDR: D802
-------------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------------

  D802  ---
  D803  RETURN

  D804  yes, update entries left counter (F058)
  D80A  back to first buffer page ($49)
  D80C  check next block pointer (F602)
  D814  if zero, directory error >>D800
  D816  BLKNUM = next directory block
  D81D  read next block <DDE1>
  D820  no errors, loop back for more >>D7E0
  D822   exit if error

          *** NO MORE FILE ENTRIES ***

  D823  free entry found in directory? (F05B)
  D826  yes >>D843
  D828  no, check pointers (F602)
  D82B  is there another block after this one? >>D832
  D830  no... >>D843
  D832  yes, free entry will be.. (F01C)
  D83B  first in that block
  D840  indicate free entry available (F05B)
  D843  find next index name <D970>
  D846   exiting with error
  D847  no more indicies in path, file not found >>D84C
  D849  else, path not found
  D84B  RETURN

  D84C  file not found error
  D84E  RETURN

          *** FOUND FILE ENTRY ***

  D84F  advance to next subdir in path <D969>
  D852  end -- save entry no. and exit >>D8C0
  D856  get type of entry
  D85A  subdir?
  D85C  no, bad path then >>D846
  D860  copy key block no...
  D862  to BLKNUM
  D865  and to current DIR block no (F01A)
  D86F  go read key block of subdirectory <DDE1>
  D872  error? >>D898
  D877  new file count (F058)
  D880  check minimum version (F621)
  D883  too new? >>D896
  D88B  count bits in reserved field of DIR hdr
  D88C  --- >>D88F
  D88F  ---
  D892  there must be 5 bits on (normally $75)
  D894  (there are) >>D89A
```

```
ProDOS MLI -- V1.0.1 -- 1 JAN 84                NEXT OBJECT ADDR: D896
-------------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------------

  D896  or else, incompatible file format
  D898  ---
  D899  RETURN

  D89A  copy DIR HDR <D8A0>
  D89D  and go scan for next level >>D7DA

D8A0 ********* COPY DIRECTORY HDR ************************************

  D8A0  Copy:
  D8A2  CREATION, VERSION, MIN_VERS, ACCESS, (F61C)
  D8A5  ENTRY_LEN, ENTRIES_PER_BLK, FILE_COUNT (F00A)
  D8AB  volume directory? (F604)
  D8B2  if so, exit now >>D8BF
  D8B6  else, copy PARENT_POINTER, (F627)
  D8B9  PARENT_ENTRY_NO., and PARENT_ENTRY_LEN (F006)
  D8BF  RETURN

D8C0 ********* SAVE DIR ENTRY NO. & BLOCK ****************************

  D8C0  compute entry number (F012)
  D8C9  save it (F01E)
  D8CE  and the block it's in (F01C)
  D8D7   exit

D8D8 ********* SEARCH ONE DIR BLOCK FOR FILE *************************

  D8D8  get entries in this block (F012)
  D8DE  $48/$49 --> first entry (D82A)
  D8E5  ---
  D8E7  skip HDR? >>D91C
  D8E9  no, non empty entry?
  D8ED  yes >>D8FC
  D8EF  no, do we need one? (F05B)
  D8F2  no >>D91C
  D8F4  yes, remember it <D8C0>
  D8F7  don't need another one now (F05B)
  D8FA  skip to next entry >>D91C
  D8FC  get length of name
  D8FE  count it (F057)
  D901  save it for loop (F078)
  D907  same len as we are wanting? (F100)
  D90A  no, skip it >>D91C
  D90C  ---
  D910  compare names (F100)
  D91A   we found it! exit
  D91B  RETURN
```

```
ProDOS MLI -- V1.0.1 -- 1 JAN 84                NEXT OBJECT ADDR: D91B
-------------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------------

  D91C  skip to next entry (F05A)
  D920  end of block? if so, exit >>D91B
  D926  bump $48/$49 by entry len
  D92D  and go check next >>D8E5

D92F ********* GET DIRECTORY DATA ************************************

  D92F  find base directory <D988>
  D932  error? >>D987
  D938  zero out my variables (F006)
  D93E  set up device number (BF30)
  D944  copy DIR HDR to my variables <D8A0>
  D94D  copy TOTAL BLOCKS from VCB (F212)
  D953  copy BIT MAP Pointer from VCB (F21A)
  D959  copy Block No. of this directory (0046)
  D95F  make second copy of file count (F013)
  D969  advance to next subdir in path <D970>
  D96C  and update index (F07A)
  D96F  RETURN

D970 ********* ADVANCE TO NEXT DIR NAME ******************************

  D970  get this DIR's index (F07A)
  D977  add len of name to move index to next name (F07A)
  D97B  still in prefix portion? >>D983
  D97D  no, now starting caller's path suffix (BF30)
  D980  save last DEVNUM accessed (F05F)
  D983  return with len of next dir in path (F100)
  D987  RETURN

D988 ********* FIND BASE DIRECTORY ***********************************

  D988  ---
  D98A  get old PFIXPTR (BF9A)
  D98D  fully qualified pathname? (F07C)
  D990  no >>D993
  D992   yes, no old PFIXPTR anymore
  D993  save old prefix index (F07B)
  D996  DEVNUM=0 (BF30)
  D999  ---

          *** SCAN VCB'S FOR A MOUNTED VOLUME ***

  D99B  scan (F200)
  D99E  got one >>D9AC
  D9A5  else, bump to next VCB
  D9A9  no mounted vols? remount them >>D9FD
```

```
ProDOS MLI -- V1.0.1 -- 1 JAN 84                NEXT OBJECT ADDR: D9A9
-------------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------------

        *** FIND LAST DIR IN PREFIX OR VOL DIR ***

 D9AC  store name length (F078)
 D9AF  same name as in pathname? (F100)
 D9B2  no -- skip it >>D9A0
 D9C0  save VCB index (F051)
 D9C3  DEVNUM = VCB's unit no. (F210)
 D9C9  BLOCK = 2 (read VOLDIR if no old PFIX)
 D9D1  get old prefix index (F07B)
 D9D4  ---
 D9D5  accumulate a new index (F07A)
 D9D8  no previous prefix? >>D9EA
 D9DB  find last name in prefix (F100)
 D9E0  read prefix directory instead of vol dir (F060)
 D9EA  read block <DDE1>
 D9ED  error? >>D9F5
 D9EF  is this the right directory? <DA91>
 D9F2  no >>D9F5
 D9F4   yes -- exit!

        *** IF NOT THERE, REMOUNT ALL VOLS ***
        *** AND CHECK THEM                 ***

 D9F5  open files? (F051)
 D9FB  yes, give up now >>DA16
 D9FD  else, (F07B)
 DA00  put back old prefix length (F07A)
 DA03  copy DVCLST from global page <DA57>
 DA09  use last device accessed first >>DA1A
 DA0B  if none, get last in my device table (BF31)
 DA16  volume not found error
 DA19  RETURN

 DA1A  ---
 DA1D  search for device in device table (F08A)
 DA25  when found, make it active device (BF30)
 DA2A  remove it from table (F08A)
 DA2D  find its VCB <DA69>
 DA30  not found? >>DA56
 DA32  volume mounted there? (F051)
 DA38  no >>DA3F
 DA3A  yes, open files here? (F211)
 DA3D  yes, skip it -- get next unit >>DA0B
 DA3F  else,
 DA41  BLKNUM = 2 (vol dir)
 DA47  read volume directory <DDE1>
 DA4A  error?  >>DA0B
 DA4C  mount volume on VCB <DAB7>
 DA4F  error? >>DA0B
 DA51  is this his chosen volume? <DA91>
```

```
ProDOS MLI -- V1.0.1 -- 1 JAN 84                NEXT OBJECT ADDR: DA54
-------------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------------

 DA54  no, try again >>DA0B
 DA56   yes, exit

DA57 ********* COPY GLBL DEVLST TO MY TABLE ***************************

 DA57  start with last device (BF31)
 DA5A  get a unit number (BF32)
 DA5F  copy it to device table (F08A)
 DA65  return count of devices (BF31)
 DA68  RETURN

DA69 ********* SCAN VCB'S FOR DEVICE NO. ******************************

 DA69  ---
 DA6D   scan VCB's for a given device number
 DA74  not it? >>DA7B
 DA76  is it, save VCB index (F051)
 DA79   and exit normally
 DA7A  RETURN

 DA7B  else, volume mounted here? (F200)
 DA7E  yes >>DA84
 DA81  no, save VCB index to empty unit (F051)
 DA84  ---
 DA86  bump to next VCB
 DA88  and go look at it >>DA6D
 DA8A   not found...
 DA8B  any free entries? if not, error >>DA8E
 DA8D   else, all is well -- return empty VCB
 DA8E  VCB table full error
 DA90  RETURN

DA91 ********* COMPARE DIR NAME WITH PATH LVL ************************

 DA91  ---
 DA96  check DIR type (F604)
 DA99  VOL DIR or SUB DIR?
 DA9B  neither >>DAA4
 DA9D  yes..
 DA9F  store len of its name (F078)
 DAA2  and go on >>DAA9
 DAA4   error exit
 DAA5  RETURN

 DAA6  compare directory names (F604)
 DAAC  no match? >>DAA4
 DAB5   they match! exit
 DAB6  RETURN
```

```
ProDOS MLI -- V1.0.1 -- 1 JAN 84                NEXT OBJECT ADDR: DAB6
-------------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------------


DAB7 ********* MOUNT NEW VOLUME ************************************

  DAB7  volume mounted? (F051)
  DABD  no, continue >>DAC4
  DABF  yes, same one as one wanted? <DB1C>
  DAC2  if so exit, else fall thru >>DB1B

DAC4 ********* SET UP VCB FRDM VOLDIR ******************************

  DAC4  zero out VCB
  DACF  is this a ProDOS volume? <D77D>
  DAD2  no -- exit >>DB1B
  DAD4  duplicate vol in VCB's? <DB3D>
  DAD7  yes -- exit with that one instead >>DB1A
  DAD9  get new volume's name length (F604)
  DAE0  add to VCB index (F051)
  DAE4  and copy to VCB name field in empty VCB (F604)
  DAEF  store in VCB name len field (F200)
  DAF2  copy DEVNUM to VCB unit field (BF30)
  DAF8  copy total blocks to VCB (F629)
  DB04  copy block no. of vol dir to VCB
  DB0E  copy bit map block no. to VCB (F627)
  DB1A   exit
  DB1B  RETURN

DB1C ********* COMPARE VDL NAMES TO MAKE ****************************
          ****** SURE THEY MATCH           ******

  DB1C  get length (F604)
  DB21  same in VCB? (F200)
  DB24  no >>DB34
  DB27  yes, add len to VCB index to point at (F050)
  DB2A  last char of name in VCB (F050)
  DB31  compare names (F200)
  DB34   SEC if no match
  DB3B   CLC if match
  DB3C  RETURN

DB3D ********* LOOK FOR DUPLICATE VDL ********************************

  DB3D  start with first VCB
  DB3F  ---
  DB40  this VCB has same name? <DB1C>
  DB43  no >>DB54
  DB45  yes, files open? (F211)
  DB48  yes >>DB5E
  DB4C  no, mark VCB empty (NAME=0) (F200)
  DB4F  (UNIT=0) (F210)
  DB52  and exit with no error >>DB5C
  DB54   else,
```

```
ProDOS MLI -- V1.0.1 -- 1 JAN 84                NEXT OBJECT ADDR: DB56
-------------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------------

  DB56  bump to next VCB
  DB5A  and loop >>DB3F
  DB5C   exit no errors
  DB5D  RETURN

  DB5E  save flag (F075)
  DB61  and VCB index of duplicate vol (F076)
  DB64   exit with error
  DB65  RETURN

DB66 ********* SEE IF A QUANTITY OF FREE  ****************************
          ****** BLOCKS IS AVAILABLE ON VOL ******

  DB66  any free blocks counted in VCB? (F051)
  DB6F  yes >>DBC3

          *** COMPUTE VCB FREE BLOCK COUNT ***

  DB71  no, how many bit map blocks are there? <DC15>
  DB74  save it (less 1) (F05C)
  DB79  zero scratch (will count free blocks) (F046)
  DB7F  no block found yet
  DB84  checkpoint bit map buffer <DD86>
  DB87  error? >>DBD7
  DB8C  BLKNUM = bit map pointer (F21A)
  DB96  read block to buffer <DDE1>
  DB99  error? >>DBD7
  DB9B  count free blocks marked <DBD8>
  DB9E  drop no. remaining to do (F05C)
  DBA1  none left? >>DBAC
  DBA3  some, BLKNUM = BLKNUM + 1
  DBA9  go process that >>DB96

  DBAC  did we find a free bit? (F051)
  DBB2  no -- volume full >>DBD4
  DBB4  save VCB bitmap block offset (F21C)
  DBB7  save free block count in VCB also (F047)
  DBC3  are there enough to satisfy request? (F214)
  DBD2   yes, exit
  DBD3  RETURN

  DBD4  volume full error
  DBD7  RETURN

DBD8 ********* SCAN AND COUNT BITMAP BLOCKS **************************

  DBD8  scan through both buffer pages
  DBDF  counting one bits <DC05>
  DBEA  ---
  DBED  found free block already? (F05B)
  DBF0  if so -- done >>DC04
```

```
ProDOS MLI -- V1.0.1 -- 1 JAN 84                NEXT OBJECT ADDR: DBF2
-------------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------------

  DBF2  any blocks found yet? (F046)
  DBF8  no >>DC04
  DBFA  yes, compute total no. of bitmap blocks <DC15>
  DBFE  less number remaining (F05C)
  DC01  gives bitmap block with first free bit (F05B)
  DC04   exit

DC05 ********* COUNT ONE BITS IN A BYTE *******************************

  DC05  shift and...
  DC08  count bits that are on (F046)
  DC10  exit when byte goes to zero
  DC14  RETURN

DC15 ********* COMPUTE NO. BITMAP BLKS -1 *****************************

  DC15  get blocks on vol count (-1) (F051)
  DC21  ---
  DC22  isolate top nibble of block count
  DC23  for bit map block count
  DC26  RETURN

DC27 ********* FREE A BLOCK ON DISK ***********************************

  DC27  save MSB (F05C)
  DC2A   and LSB
  DC2E  block number passed too big for (F213)
  DC31  volume size? (F05C)
  DC35  yes, error >>DCA5
  DC38  no, get bit position for block no.
  DC3E  save it (F05B)
  DC42  divide block no. by 8 (F05C)
  DC45  giving byte offset as remainder
  DC4E  save byte offset (F062)
  DC51  make quotient/2 into block index (F05C)
  DC54  remember which page in that block (F064)
  DC57  read bit map block (after checkpoint) <DD57>
  DC5A  error? >>DCA4
  DC5C  are we at proper block of bitmap yet? (F069)
  DC62  yes! >>DC7A
  DC64  no -- checkpoint <DD86>
  DC67  error? >>DCA4
  DC69  indicate block wanted in VCB (F05C)
  DC72  DEVNUM of bitmap (F066)
  DC75  read actual block directly <DD97>
  DC78  error? >>DCA4
  DC7A  get byte offset into page (F062)
  DC7D  which page? (F064)
  DC80  get bit pattern to set (F05B)
  DC83  page 0? >>DC8D
  DC85  no, turn bit on in page 1 (F500)
```

```
ProDOS MLI -- V1.0.1 -- 1 JAN 84                NEXT OBJECT ADDR: DC8B
-------------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------------

  DC8B  and continue >>DC93
  DC8D  turn bit on in page 0 (F400)
  DC93  mark bitmap needs checkpoint
  DC9B  count block freed (F082)
  DCA3   exit normally
  DCA4  RETURN

  DCA5  bad bitmap error
  DCA8  RETURN

DCA9 ********* FIND A FREE DISK BLOCK AND ****************************
          ****** AND ALLOCATE IT            ******

  DCA9  go read bitmap <DD57>
  DCAC  error? >>DCD1
  DCAE  first page 0
  DCB3  scan 1st page of bitmap for free block(s) (F400)
  DCBB  bump to page 1 of buffer (F064)
  DCBE  bump page offset (F063)
  DCC1  scan 2nd page too (F500)
  DCC9  bump page (F063)
  DCCC  get next block <DD35>
  DCCF  continue >>DCAE
  DCD1   error exit

  DCD2  save byte index (F062)
  DCD5  shift combination of page no. and (F063)
  DCD8  byte offset left 3 bits to make (F047)
  DCDB   room for bit position.
  DCEA  depending on buffer page ...  (F064)
  DCEF  reload bit pattern from page 0... (F500)
  DCF4  or page 1 (F400)
  DCF7  shift bit pattern, bumping block no. LSB
  DCF8  until a one bit is found >>DCFD
  DCFD  then shift it back the way it was
  DCFE  (with that bit turned off) >>DCFD
  DD00  store LSB of block no. (F046)
  DD03  store updated byte back in proper page (F064)
  DD10  indicate bitmap needs checkpoint
  DD18  one less block available in VCB (F051)
  DD2D  ---
  DD2E  return with new block no. (F046)
  DD34  RETURN

DD35 ********* GET NEXT BITMAP BLOCK **********************************

  DD35  use blocks of vol to compute (F051)
  DD38  number of blocks in bitmap (F213)
  DD3F  just scanned last block? (F21C)
  DD42  yes, no space >>DD53
  DD44  no, get next block (F21C)
```

```
ProDOS MLI -- V1.Ø.1 -- 1 JAN 84                NEXT OBJECT ADDR: DD4D
-------------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------------

  DD4D  checkpoint old one <DD86>
  DD5Ø  go read block >>DD57

  DD53  disk full error
  DD56  RETURN

DD57 ********* READ BITMAP BLOCK ************************************

  DD57  have we read bitmap for this unit yet? (FØ51)
  DD6Ø  yes >>DD7Ø
  DD62  no, checkpoint bitmap of some other unit  <DD86>
  DD65  error? >>DD85
  DD6A  get new bitmap unit no. (F21Ø)
  DD7Ø  was bitmap modified? (FØ65)
  DD73  yes >>DD7A
  DD75  no, read it <DD97>
  DD78  error? >>DD85
  DD7A  save bitmap block offset times 2 (FØ51)
  DD7D  (page number) (F21C)
  DD84   exit
  DD85  RETURN

DD86 ********* CHECKPOINT VOLUME BITMAP *****************************

  DD86  ---
  DD87  needs checkpoint? (FØ65)
  DD8A  no >>DD85
  DD8C  yes, write it <DDD9>
  DD8F  error? >>DD85
  DD91  doesn't need checkpoint now
  DD96   exit

DD97 ********* READ BITMAP ******************************************

  DD97  save DEVNUM (FØ66)
  DD9A  copy block offset wanted (FØ51)
  DDA4  BITMAP BLOCK = BITMAP PTR + BLOCK OFFSET (F21A)
  DDB2  set up read command

          *** READ OR WRITE BITMAP ***

  DDB4  save I/O command
  DDBA  device = bitmap device (FØ66)
  DDCØ  block = bitmap block (FØ67)
  DDCA  point to bitmap buffer (DC8F)
  DDCD  do the I/O <DDE8>
  DDD2  restore old DEVNUM (BF3Ø)
  DDD5  ok? >>DDD8
  DDD7   no, error exit
  DDD8  RETURN
```

```
ProDOS MLI -- V1.Ø.1 -- 1 JAN 84                NEXT OBJECT ADDR: DDD8
-------------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------------


DDD9 ********* WRITE BITMAP ******************************************

  DDD9  set up write command
  DDDB  and go do it >>DDB4

DDDD ********* WRITE BLOCK *******************************************

  DDDD  set up write command
  DDDF  and go do it >>DDE3

DDE1 ********* READ BLOCK ********************************************

  DDE1  set up read command

DDE3 ********* READ OR WRITE BLOCK ***********************************

  DDE3  save I/O command
  DDE5  where is my buffer? (D82A)
  DDE8   save flags
  DDE9   and disable
  DDEC  Set low byte of Buffer pointer
  DDEE  to zero
  DDFØ  Initialize Global Page System error to Ø (BFØF)
  DDF3  set I/O transfer occured flag
  DDF8  set unit to do I/O on (BF3Ø)
  DDFD  do block I/O <DØDA>
  DEØØ  error? >>DE05
  DEØ2   no errors, restore things and exit
  DEØ4  RETURN

  DEØ5   error exit
  DEØ7  RETURN

DEØ8 *************************************************************
          ****** MLI GET_MARK CALL ******
          ******************************

  DEØ8  copy mark to caller's list from FCB (F052)
  DE18   exit with no errors
  DE19  RETURN

  DE1A  bad position error
  DE1D  RETURN

DE1E *************************************************************
          ****** MLI SET_MARK CALL ******
          ******************************
```

```
ProDOS MLI -- V1.0.1 -- 1 JAN 84                NEXT OBJECT ADDR: DE1E
-------------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------------

 DE1E  set up to...
 DE26  copy user's mark to temporary
 DE28  new mark variable (F068)
 DE2D  make sure it will not exceed EOF (F315)
 DE32  else, error >>DE1A
 DE35  ---

         *** STILL IN SAME DATA BLOCK? ***

 DE3B  get old mark (F052)
 DE3E  find its block no. (*2) (F313)
 DE46  compute distance in pages from old mark's (F06B)
 DE4A  block to new mark (F046)
 DE50  earlier -- need new data block >>DE61
 DE54  too far forward -- need new block >>DE61
 DE59  MSB's match? (F314)
 DE5E  then mark is still in this block >>DF7C

 DE61  check storage type (F307)
 DE64  zero? >>DE6D
 DE66  seedling, sapling or tree?
 DE6A  no, special handling for DIR files >>DFAE

 DE6D  stomp on FCB2's mark??? (F300+$52)
 DE6F  (this should never happen anyway) (F300)
 DE72  and return with bad REFNUM error
 DE75  RETURN

         *** NEED DIFFERENT DATA BLOCK ***

 DE76  copy storage type (F307)
 DE7C  old data block needs writing? (F308)
 DE81  no >>DE88
 DE83  yes, do so <E087>
 DE86  error? >>DEF1
 DE88  see if new mark is outside the range of  (F052)
 DE8B  the current index block (F314)
 DE9A  yes >>DEBA
 DE9E  yes >>DEBA
 DEA0  no, same index block (F056)
 DEA3   check storage type
 DEA4  sapling or tree are ok >>DF20

         *** SEEDLING ***

 DEA6  seedling, check position (F06B)
 DEA9  if position is outside of block 0..
 DEAD  promote to sapling >>DF0E
 DEAF  else, (F30C)
 DEB7  go get key block (seedling data block) >>DF72
```

```
ProDOS MLI -- V1.0.1 -- 1 JAN 84                NEXT OBJECT ADDR: DEB7
-------------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------------

         *** NEED TO CHANGE DATA BLOCKS ***

 DEBA  does old index block need dumping? (F308)
 DEBF  no >>DEC6
 DEC1  yes, do so <E09B>
 DEC4  error? >>DEF1
 DEC6  check storage type (F056)
 DEC9  tree file?
 DECB  yes >>DEF3
 DECD  no, sapling (F06C)
 DED2  is position in first index block?
 DED5  no, need master index, subindex and data >>DF39
 DED7  yes, first index, reset flags <DFA2>
 DEDA   is this a seedling?
 DEDB  if so, see if in first block >>DEA6

         *** SAPLING ***

 DEDD  no, sapling, read its only index block <E02E>
 DEE0  error? >>DEF1
 DEE5  set block no. of index block
 DEEF  and continue below >>DF20
 DEF1   error exit
 DEF2  RETURN

         *** TREE FILE/NEED ANOTHER INDEX BLOCK ***

 DEF3  reset flags  <DFA2>
 DEF6  read master index block <E02E>
 DEF9  error? >>DEF1
 DEFB  make index into block from (F06C)
 DEFE  MSB_of_position/2
 DF04  is there a subindex there?
 DF06  yes! >>DF13
 DF0C  no, fall thru to make one

         *** GET NEW INDEX BLOCK ***

 DF0E  need an index and data block
 DF10  go allocate them >>DF39

 DF13  set up block no. of subindex
 DF1B  read it <E010>
 DF1E  error? >>DEF1

         *** SAPLING/TREE - THIS INDEX BLOCK ***
```

```
ProDDS MLI -- V1.0.1 -- 1 JAN 84                NEXT OBJECT ADDR: DF20
-------------------------------------------------------------------------
ADDR    DESCRIPTIDN/CDNTENTS
-------------------------------------------------------------------------

  DF20  make block no. out of position (F06C)
  DF29  use as an index to examine index block
  DF2B  entry
  DF31  if its zero...
  DF35  need new data block
  DF39  set flags for what to allocate (F052)
  DF42  new index block being created?
  DF44  zero data block in any case <DF5A>
  DF47  if not index block that's it >>DF7C
  DF49  else,
  DF50  zero out index block I/D buffer
  DF57  and continue >>DF7C

DF5A ********* ZERD OUT DATA BLK I/D BUFFER **************************

  DF5A  ---
  DF5D  zero both pages of buffer
  DF64  ---
  DF6B  RETURN

DF6C ********* READ FILE DATA BLOCK **********************************

  DF6C  set block no. LSB
  DF6E  copy MSB from index entry
  DF72  ---
  DF74  read new data block <DFF7>
  DF77  error? >>DFA1
  DF79  reset block allocation flags <DFA2>

          *** GOT DATA BLDCK WANTED ***

  DF7C  ---
  DF83  save previous mark in my variables (F312)
  DF89  set new mark in the FCB (F06A)
  DF94  ($4A/$4B --> data block buffer)
  DF96  $4C/$4D --> start of the page in
  DF98  the data block buffer which contains (F06B)
  DF9B  the mark.
  DFA1   exit

DFA2 ********* RESET BLOCK ALLDC FLAGS ******************************

  DFA2  get flags (F052)
  DFA8  turn off low 3 bits (allocate no new
  DFAA  blocks to file) (F308)
  DFAD  RETURN
```

```
ProDDS MLI -- V1.0.1 -- 1 JAN 84                NEXT DBJECT ADDR: DFAD
-------------------------------------------------------------------------
ADDR    DESCRIPTIDN/CONTENTS
-------------------------------------------------------------------------

DFAE ********* SET DIR FILE PDSITION *********************************

  DFAE  DIR file?
  DFB0  yes! >>DFB7
  DFB2  no, bad storage type error
  DFB4  got to SYSERR <BF09>
  DFB7  else, get page distance (F046)
  DFBA  make it into blocks (divide by 2)
  DFC1  new position beyond old? (F06B)
  DFC4  yes >>DFD4
  DFC6  else, use previous mark
  DFC8  copy to BLKNUM <DFE2>
  DFCB  error? >>DFF1
  DFCD  count it (F05A)
  DFD0  more to skip? >>DFC6
  DFD2  no, got it >>DF7C
  DFD4  use next block pointer in DIR block
  DFD6  copy to BLKNUM <DFE2>
  DFD9  error? >>DFF1
  DFDB  count it (F05A)
  DFDE  more to skip >>DFD4
  DFE0  got it now! >>DF7C

          *** CDPY LINK TD BLKNUM ***

  DFE2  copy block number link
  DFE4  to BLKNUM
  DFE7  if non zero,
  DFED  then go read block. >>DFF3
  DFEF  else, EOF error
  DFF1  ---
  DFF2  RETURN

DFF3 ********* READ FILE BLDCK ***************************************

  DFF3  set block number to read
  DFF7  store read I/D command
  DFFB  read to $48/$49 buffer
  DFFD  read the block <E054>
  E000  error? >>E00F
  E005  copy block no. just read to FCB
  E00F   exit

E010 ********* READ SUB-INDEX BLOCK **********************************

  E010  set read I/D command
  E014  read to $48/$49 buffer
  E016  read the block <E054>
  E019  error? >>E029
  E01E  save BLKNUM in FCB as current index
```

```
ProDOS MLI -- V1.0.1 -- 1 JAN 84                NEXT OBJECT ADDR: E020
-------------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------------

  E020  block. (F30E)
  E029   exit

E02A ********* WRITE KEY INDEX BLOCK *********************************

  E02A  set write I/O command
  E02C  and go do the I/O >>E030

E02E ********* READ KEY INDEX BLOCK **********************************

  E02E  set read I/O command
  E030   common code, save command
  E033  block no. is key block in FCB (F052)
  E038  use $48/$49 buffer

          *** I/O BLOCK ***

  E03A  set I/O command
  E03C  and block no. (F300)
  E046  must be non-zero block number
  E04A  or horrible death!
  E04F  fall through to read/write block (F301)

          *** SET UP AND DO FILE BLOCK I/O ***

  E054   (xreg = buff ptr in zero page)
  E055   disable
  E056  set up buffer pointer
  E061  get DEVNUM from FCB (F301)
  E067  set I/O transfer has occured flag
  E06C  set unit no. from DEVNUM (BF30)
  E071  no errors have occured yet
  E076  do block I/O <D0DA>
  E079  error? >>E07E
  E07B   no, exit normally
  E07D  RETURN

  E07E   else, exit with error
  E080  RETURN

E081 ********* CHECKPOINT BITMAP & KEY BLOCK *************************

  E081  checkpoint bitmap buffer <DD86>
  E084  go write key block for file >>E02A

E087 ********* CHECKPOINT DATA BLOCK BUFFER **************************
```

```
ProDOS MLI -- V1.0.1 -- 1 JAN 84                NEXT OBJECT ADDR: E087
-------------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------------

  E087  buffer pointer at $4A/$4B
  E089  point to block no. in FCB
  E091  go write buffer to disk <E03A>
  E094  error? >>E0B8
  E098  go turn off $40 flag in FCB and exit >>E0AF

E09B ********* CHECKPOINT INDEX BLOCK BUFFER *************************

  E09B  checkpoint volume bitmap <DD86>
  E09E  use $48/$49 buffer
  E0A0  block no. is current index block in FCB
  E0A6  set to write
  E0A8  go write it to disk <E03A>
  E0AB  error? >>E0B8
  E0AD  no longer needs checkpoint
  E0AF  set flags accordingly (F052)
  E0B8   and exit

E0B9 ************************************************************************
          ****** MLI OPEN CALL ******
          **************************

  E0B9  search path for file <D798>
  E0BC  found it? >>E0C2
  E0BE  no, bad path error
  E0C0  exit >>E0C9
  E0C2  else, see if FCB already open on file <E1A9>
  E0C5  for write. if not, continue. >>E0CB
  E0C7  else, file already open error
  E0C9  ---
  E0CA  RETURN

  E0CB  get FCB index (F052)
  E0D1  free FCB found? >>E0D7
  E0D3  no, all FCB's in use error
  E0D6  RETURN

  E0D7  zero out unused FCB
  E0E2  copy file ID fields to FCB
  E0E5  (DEVNUM, DIR HDR BLK, DIR BLK, (F052)
  E0E8   DIR ENTRY NO.)
  E0F3  isolate storage type (F01F)
  E0FB  and copy to FCB (F307)
  E0FE  get access (F03D)
  E103  DIR file?
  E105  no >>E109
  E107  yes, we are only reading (I hope)
  E109  update access flag in FCB (F309)
  E10E  write protected? >>E115
  E110  no, another FCB open on this file? (F057)
```

```
ProDOS MLI -- V1.0.1 -- 1 JAN 84                NEXT OBJECT ADDR: E113
-------------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------------

  E113  yes, no touchie >>E0C7
  E115  else, check file min_version (F03C)
  E118  against global page version (BFFF)
  E11F  if bad, unsupported version error
  E122  RETURN

  E123  storage type must be < $4
  E127  or equal to $D
  E129  else, compatibility error >>E11F
  E12B  ---
  E12D  copy key block, blocks used, and
  E12F  EOF mark to FCB (F052)
  E13F  BLKNUM = key block number
  E144  store REFNUM in FCB (F05A)
  E14A  go check and assign I/O buffer <EDD7>
  E14D  error? >>E173
  E14F  go find VCB and set buff ptrs <D3E0>
  E152  set current level in FCB (BF94)
  E158  seedling, sapling or tree? (F307)
  E15D  no, skip next stuff >>E18A
  E15F  yes, make current mark in FCB outside
  E161  first index block to force a read of all (F314)
  E164  index blocks and BLOCK 0.
  E168  zero mark wanted, however (F06A)
  E16E  go set mark to zero <DE3B>
  E171  ok? >>E18F
  E173   no, save the error code
  E177  got and I/O buffer? (F30B)
  E17A  no >>E182
  E17C  yes, free it <EE34>
  E182  mark FCB not in use
  E188   exit with error
  E189  RETURN

  E18A  else, read key block to I/O buffer <DFF7>
  E18D  error? >>E173
  E18F  bump open file count in VCB (F051)
  E195  indicate files are open in VCB (F211)
  E19D  put REF NUM in caller's parmlist (F052)
  E1A7   exit with no errors
  E1A8  RETURN

E1A9 ********* FIND A FCB *******************************************

  E1A9  clear flags and index byte
  E1B4  ---
  E1B5  found a free FCB yet? (F053)
  E1B8  yes >>E1BD
  E1BA  no, bump entry count (F05A)
  E1BD  FCB in use? (F300)
  E1C0  yes >>E1CF
```

```
ProDOS MLI -- V1.0.1 -- 1 JAN 84                NEXT OBJECT ADDR: E1C2
-------------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------------

  E1C2   no,
  E1C5  save index to free FCB (F052)
  E1C8  flag that we found one
  E1CD  and skip this FCB >>E1ED
  E1CF  ---
  E1D5  compare file ID's to see if this FCB (F300)
  E1D8  is open on the requested file. (F018)
  E1DB  no match? >>E1ED
  E1E1  indicate FCB already open on file (F057)
  E1E4  write enabled? (F309)
  E1E9  if not, allow multiple open access to file >>E1ED
  E1EB   else, error exit
  E1EC  RETURN

  E1ED   return index to start of FCB
  E1F1  bump to next FCB
  E1F3  and loop >>E1B4
  E1F5   when done, exit normally
  E1F6  RETURN

E1F7 ************************************************************
       ****** MLI READ CALL ******
       **************************

  E1F7  point to data buffer <E403>
  E1FA  copy request length <E3E8>
  E1FD   save access
  E1FE  set up marks <E415>
  E202  read access permitted?
  E204  yes >>E20A
  E206  no, access error
  E20A  will we read past EOF? >>E231
  E20C  yes, (F052)
  E20F  LENGTH = EOF - current_mark (F315)
  E227  are we already at EOF? (F09A)
  E22A  no >>E23C
  E22C  yes, EOF error

  E231  else, zero length request? (F09A)
  E237  no >>E23C
  E239  yes, set mark and exit >>E2EF

  E23C  validity check data buffer <EE6C>
  E23F  no good? >>E22E
  E241  ok, get storage type for file <E40E>
  E244  standard kind of file?
  E246  yes >>E24B
  E248  no, DIR file >>E3B1
```

```
ProDOS MLI -- V1.0.1 -- 1 JAN 84                NEXT OBJECT ADDR: E248
-------------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------------

  E24B  else, set mark (to read proper buffers) <DE3B>
  E24E  error? >>E22E
  E250  set up buffer indexing <E306>
  E253  move all that can be moved out of data buff <E330>
  E256  newline or len=0: exit now!  >>E239
  E258  newline enabled? continue block by block >>E24B
  E25A  at least 1 block's worth left to be read? (F06E)
  E25E  if not, never mind >>E24B
  E260  if so, store block count wanted (F06F)
  E263  get FCB flags <E7FC>
  E266  data block modified?
  E268  yes, continue block by block for now >>E24B

          *** FAST DIRECT READ ROUTINE ***

  E26A  signal no read occured yet (F072)
  E26D  read directly into caller's data buffer
  E275  set mark/read data block to caller's buff <DE3B>
  E278  error? >>E2E3
  E27A  bump buffer pointer to next location
  E27E  drop length remaining by 512 bytes (F06E)
  E284  bump mark (F06B)
  E28C  and mark's MSB as necessary (F06C)
  E28F  check if we are out of index block (F06C)
  E295  drop counter of multi-blocks (F06F)
  E298  and keep on >>E2A7
  E29A  end of multi-block read, put ptrs back <E3A3>
  E29D  more to read? (F06D)
  E2A3  no, exit through finish-up >>E2EF
  E2A5  yes, conventional block by block read then >>E24B

  E2A7  crossed index block? go do set mark >>E275
  E2A9  make index block offset from mark (F06C)
  E2B2  BLKNUM = next block in index block
  E2B8  zero entry?
  E2C0  if so, no direct read can occur until next (F072)
  E2C3  set-mark/read >>E2C8
  E2C5  get MSB of BLKNUM
  E2C8  (put index ptr back)
  E2CC  finish setting BLKNUM MSB
  E2CE  if no read occured within setmark, (F072)
  E2D1  go back to setmark call >>E275
  E2D5   disable
  E2D6  do I/O to caller's buffer directly
  E2DA  do block I/O directly <D0DA>
  E2DD  error? >>E2E2
  E2E0  go back for more >>E27A
```

```
ProDOS MLI -- V1.0.1 -- 1 JAN 84                NEXT OBJECT ADDR: E2E0
-------------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------------

          *** ERROR CLEANUP ***

  E2E2  ---
  E2E3  ---
  E2E4  set buffer ptrs/VCB <E3A3>
  E2E8  ---
  E2E9  finish up I/O <E2EF>
  E2ED   exit with error
  E2EE  RETURN

E2EF ********* I/O FINISH UP ***************************************

  E2EF  ---
  E2F2  return actual length read in caller's list (F09A)
  E303  and exit by setting new mark >>DE3B

E306 ********* SET UP BUFFER INDEXING *********************************

  E306  ---
  E30A  back up pointer to data buffer by an
  E30C  amount equal to the LSB of the mark (F06A)
  E30F  (which makes indexing easier)
  E315  newline mode enabled? (F31F)
  E319  no, CLC >>E325
  E31B   yes, SEC
  E31C  copy newline mask (F071)
  E31F  and newline character (F30A)
  E325  first char index is LSB of mark in YREG (F06A)
  E328  $4C/$4D --> page containing mark
  E32C  request count LSB in XREG (F06D)
  E32F   exit

E330 ********* COPY FROM I/O BLOCK BUFF ******************************
          ****** TO DATA BUFFER          ******
          EXITS IF:   LENGTH GOES TO ZERO
                      NEXT BLOCK IS NEEDED
                      NEWLINE IS FOUND
          ON EXIT: OVERFLOW FLAG SET IF DONE
                   OVERFLOW ZERO IF NEXT BLOCK NEEDED

  E330  ---
  E331  partial page to move? >>E33B
  E333  no, any full pages left? (F06E)
  E336  no, read complete >>E38A
  E338  yes, drop MSB of request length (F06E)
  E33B  ---
  E33C  copy one byte $4C --> $4E
  E341  end of requested chunk? >>E35E
  E343  no, newline enabled? >>E373
  E345  ---
```

```
ProDOS MLI -- V1.0.1 -- 1 JAN 84                NEXT OBJECT ADDR: E347
-------------------------------------------------------------------------
ADDR    DESCRIPTIDN/CONTENTS
-------------------------------------------------------------------------

  E347  no, loop for more >>E33C
  E349  end of page, bump pointers
  E34D  bump new mark (F06B)
  E355  finished first page of block buffer?
  E359  if so, continue >>E33C
  E35C  no, need another block from disk >>E38D
  E35E  another page in request length? (F06E)
  E361  no >>E37D
  E364  more in this block-page? >>E36C
  E366  no, on last page of block?
  E36A  no >>E36F
  E36C  yes, drop request len by one page (F06E)
  E36F   back up to next byte again
  E370  go copy next page >>E343

  E373  check for newline
  E37B  not it, never mind! >>E345
  E37D   else, were we done with page?
  E37E  no >>E38A
  E380  yes, bump pointer
  E382  and mark (F06B)
  E38A  set overflow flag (read completed) (E3A2)

  E38D  update mark LSB (F06A)
  E392   bump request count if necessary
  E393  update count LSB (F06D)
  E399  point beyond data in caller's buffer
  E3A1  ---
  E3A2   and exit

E3A3 ********* CLEANUP AFTER DIRECT I/O ******************************

  E3A3  restore caller's data buffer pointer
  E3AE  go set buffers/find VCB and exit >>D3E0

E3B1 ********* DIRECTORY FILE READ ***********************************

  E3B1  set mark/read <DE3B>
  E3B4  error? >>E3E5
  E3B6  set up buffer indexing <E306>
  E3B9  move data from I/O buffer <E330>
  E3BC  need next block? >>E3B1
  E3BE  no, finish up I/O <E2EF>
  E3C1  ok? exit >>E3E3
  E3C3  not ok. EOF error?
  E3C6  no, out now >>E3E4
  E3C8  yes, point beyond EOF anyway? <DF7C>
  E3CB  zero out data block I/O buffer <DF5A>
  E3D3  dummy up an empty DIR block with previous (F310)
  E3D6  pointer and no forward pointer in I/O
  E3D8  buffer.
```

```
ProDOS MLI -- V1.0.1 -- 1 JAN 84                NEXT OBJECT ADDR: E3DA
-------------------------------------------------------------------------
ADDR    DESCRIPTION/CDNTENTS
-------------------------------------------------------------------------

  E3DA  zero out current block no. (F310)
  E3E3   return to caller
  E3E4  RETURN

  E3E5  finish up and error exit >>E2E8

E3E8 ********* COPY CALLER'S I/O LENGTH ******************************

  E3E8  copy request length to LENGTH and
  E3EA  a temporary variable
  E3FB  pick up ACCESS flags for file (F052)
  E401   exit to caller
  E402  RETURN

E403 ********* POINT $4E/$4F TO CALLER'S ****************************
        ****** DATA BUFFER                 ******

  E403  set up pointer
  E40E  YREG --> FCB (F052)
  E411  AREG = storage type (F307)
  E414   exit

E415 ********* COPY FILE MARK AND COMPUTE ***************************
        ****** AND COMPARE END MARK        ******

  E415  ---
  E41B  copy file mark (F312)
  E421  and set previous mark also (F04D)
  E424  add length giving new mark in scratch area (F09A)
  E42B  (3 byte addition)
  E433  will new mark exceed EOF? (F046)
  E441   return with carry set accordingly

E442 ********* SET NEW MARK & EOF ************************************

  E442  set up indexes <E474>
  E445  set new EOF in FCB (F04A)
  E44B  and new mark (F04D)
  E451  save new mark in scratch variable too (F046)
  E458  does mark exceed EOF? <E474>
  E45B  if so, we must extend EOF <E433>
  E461  save old EOF (F315)
  E469  set new EDF to mark if necessary (F046)
  E46F  ---
  E473   exit

  E474  subroutine to set 3 byte indexes
  E47B  RETURN
```

```
ProDOS MLI -- V1.0.1 -- 1 JAN 84                NEXT OBJECT ADDR: E47B
-------------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------------

E47C ************************************************************
         ****** MLI WRITE CALL ******
         ****************************

  E47C  copy request length <E3E8>
  E480  copy file mark <E415>
  E483  extend EOF if needed <E45E>
  E487  write access enabled?
  E489  yes >>E48F
  E48B  no, access error
  E48F  check status of this device <E64E>
  E492  error? >>E4CF
  E494  request length = 0? (F09A)
  E49A  no >>E49F
  E49C  yes, exit through finish-up >>E2EF

  E49F  find caller's data buffer <E403>
  E4A2  check storage type
  E4A4  if DIR file, error >>E48B
  E4A6  set mark/read blocks <DE3B>
  E4A9  error? >>E4CF
  E4AB  get FCB flags <E7FC>
  E4AE  any new blocks needed?
  E4B0  no >>E514
  E4B2  yes, allocating them
  E4B4  ---
  E4B5  count number of blocks needed
  E4B8  store number needed (F054)
  E4BE  see if the blocks are available <DB66>
  E4C1  no, disk full >>E4CF
  E4C3  yes, get FCB flags <E7FC>
  E4C6  master index block needed?
  E4C8  no >>E4D7
  E4CA  yes, go add it <E58F>
  E4CD  and go on if no errors >>E4E3

  E4CF   error,
  E4D0  set new mark/EOF <E442>
  E4D4  and finish I/O, exit with error >>E2E8

  E4D7  check FCB flags again <E7FC>
  E4DA  need sub-index block?
  E4DC  no >>E4E3
  E4DE  yes, go do it  <E5DA>
  E4E1  error? >>E4CF
  E4E3  buy a new block for data <E62E>
  E4E6  error? >>E4CF
  E4E8  get FCB flags <E7FC>
  E4EB  indicate index buffer changed
  E4ED  no new blocks needed now
```

```
ProDOS MLI -- V1.0.1 -- 1 JAN 84                NEXT OBJECT ADDR: E4EF
-------------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------------

  E4EF  update FCB flags (F308)
  E4F5  make index block offset from mark
  E4FD  store new block no. in index block (F047)
  E50A  and store it as current data block (F052)
  E514  set up buffer indexing <E306>
  E517  start writing <E51F>
  E51A  go see if more blocks are needed >>E4A6
  E51C  I/O finish up when done >>E2EF

E51F ********* COPY WRITE DATA TO I/O BLOCK ***************************

  E51F  ---
  E522  lower request count by 1 (F06E)
  E52A  ---
  E52B  copy partial page from caller's data
  E52D  to I/O block buffer
  E532  ---
  E535  next page in caller's area
  E539  bump mark by $100 (F06B)
  E541  still in same I/O block page?
  E545  yes >>E52A
  E548  no, clear overflow (I/O incomplete) >>E56F

  E54A  any complete pages left to write? (F06E)
  E54D  no >>E55F
  E54F   yes, more in this page?
  E550  yes >>E558
  E552  no, first block-page?
  E556  no  >>E55B
  E558  yes, one less complete page to do (F06E)
  E55B   readjust index
  E55C  continue with full page >>E532

  E55F  ---
  E560  a few bytes left to write? >>E56C
  E562  no, bump data buffer by $100
  E564  and mark (F06B)
  E56C  set overflow (I/O complete) (E3A2)
  E56F  store LSB of mark (F06A)
  E572  and of request count (F06D)
  E576  indicate data block modifed <E7FC>
  E579  and DIR entry needs update
  E57F  advance pointer into caller's buffer (F06A)
  E58A  set FCB flag to indicate write occured <EC50>
  E58E   exit
```

```
ProDOS MLI -- V1.0.1 -- 1 JAN 84                NEXT OBJECT ADDR: E58F
-------------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------------

E58F ********* ADD NEW MASTER INDEX BLOCK ****************************
         (MAKE A TREE FILE)

  E58F  add higher level <E5E7>
  E592  error? >>E5E6
  E594  get storage type <E40E>
  E597  tree?
  E599  yes >>E5A0
  E59B  no, add another level <E5E7>
  E59E  error? >>E5E6
  E5A0  buy another block <E62E>
  E5A3  error? >>E5E6
  E5A5  make offset into current index block (F06C)
  E5A8  from current mark
  E5AA  point index to new block (F046)
  E5B9  also save as current data block (F052)
  E5C3  checkpoint bitmap & key block <E081>
  E5C6  error? >>E5E6
  E5CB  zero out new index block
  E5D2  ---
  E5D9   and exit

E5DA ********* ADD NEW INDEX BLOCK *********************************

  E5DA  check storage type <E40E>
  E5DF  seedling? >>E5E7
  E5E1  no, read key index block <E02E>
  E5E4  and go add data block >>E5A0
  E5E6   exit if error occurs

          *** ADD A HIGHER INDEX LEVEL TO FILE ***

  E5E7  buy a block <E62E>
  E5EA  error? >>E62D
  E5EF  save old key block number (F30C)
  E5F7  make new block the key block (F30C)
  E604  and current index block in FCB (F30F)
  E60D  store pointer to old key block
  E610  in first position of new index
  E617  checkpoint bitmap and new key block <E081>
  E61A  error? >>E62D
  E61C  get storage type <E40E>
  E621  upgrade it to next higher type (F307)
  E624  indicate DIR entry needs update (F308)
  E62D   exit
```

```
ProDOS MLI -- V1.0.1 -- 1 JAN 84                NEXT OBJECT ADDR: E62E
-------------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------------

E62E ********* BUY A DISK BLOCK ***************************************

  E62E  allocate a disk block <DCA9>
  E631  error? >>E64D
  E633  get FCB flags <E7FC>
  E636  indicate DIR entry needs update
  E63F  add 1 to blocks in use for file
  E64C  ---
  E64D   exit

E64E ********* DO STATUS IF NO I/O YET ********************************

  E64E  get FCB flags <E7FC>
  E651  any buffers in use? (I/O activity)
  E653  if so, assume its ok >>E64C
  E655  no, (F301)
  E658  select new device (BF30)

          *** STATUS CALL ***

  E65B  Save Unit Number
  E65D  Save Block Number on stack
  E663  Indicate Status call
  E667  Indicate Block 0
  E66B  Go do I/O <D0DA>
  E66E   Restore Block Number to original value
  E676   Exit

E677 *************************************************************************
          ****** MLI CLOSE CALL ******
          ****************************

  E677  check REF NUM
  E67B  specific close? >>E6B2

          *** CLOSE ALL OPEN FILES ***

  E67D  no errors yet (F07E)
  E682  store FCB index (F052)
  E686  get its level (F31B)
  E689  if below system LEVEL, skip it (BF94)
  E68C  yes, skip it >>E6A3
  E68E  no, active FCB? (F300)
  E691  no >>E6A3
  E693  yes, flush it and update directory <E714>
  E696  error? >>E6E5
  E698  no, close specific FCB  <E6B7>
  E69D  is this a close-all?
  E69F  yes, ignore errors >>E6A3
  E6A1  no, stop on error >>E6E5
  E6A3  bump FCB index to next one (F052)
```

```
ProDOS MLI -- V1.0.1 -- 1 JAN 84                NEXT OBJECT ADDR: E6A9
-------------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------------

  E6A9  and continue >>E682
  E6AB  when done, load error number (F07E)
  E6B1   and exit

          *** CLOSE SPECIFIC FILE ***

  E6B2  flush it <E71C>
  E6B5  error? >>E6E5
  E6B7  get buffer number (F052)
  E6BD  free its pages <EE34>
  E6C0  error? >>E6E5
  E6C2  release FCB
  E6CA  set DEVNUM (F301)
  E6D0  find VCB for device <DA69>
  E6D3  decrement count of open files in VCB (F051)
  E6D9  some are open... >>E6E3
  E6DB  if all are closed, turn off (F211)
  E6DE  "files open" flag
  E6E3  ---
  E6E4   exit

  E6E5  jump to handle close error >>E7ED

E6E8 ************************************************************
          ****** MLI FLUSH CALL ******
          ****************************

  E6E8  flush specific file?
  E6EC  yes >>E71C
  E6EE  no, clear flush-all error code (F07E)
  E6F1  do all FCBs
  E6F3  set FCB index for next FCB (F052)
  E6F7  is this file open? (F300)
  E6FA  no >>E701
  E6FC  yes, flush it <E714>
  E6FF  error? >>E711
  E701  bump to next FCB (F052)
  E707  and go flush it too >>E6F3
  E709  ---
  E70A  return with error code if any (F07E)
  E710  RETURN

  E711  ---

E714 ********** FLUSH A FILE & UPDATE DIRECTORY **********************

  E714  find buffer/VCB <D3E0>
  E717  no error? >>E726
  E719  error - exit >>E7ED
```

```
ProDOS MLI -- V1.0.1 -- 1 JAN 84                NEXT OBJECT ADDR: E719
-------------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------------

  E71C  zero out close-all error
  E721  validity check REF NUM <D3C5>
  E724  error? >>E711
  E726  is write access allowed? (F309)
  E72B  no, exit >>E709
  E72D  has a write occured since last flush? (F31C)
  E730  yes >>E739
  E732  no, <E7FC>
  E735  does anything need flushing anyway?
  E737  no, then exit now >>E709
  E739  else, get FCB flags <E7FC>
  E73C  has data buffer changed?
  E73E  no >>E745
  E740  yes, checkpoint it <E087>
  E743  error? >>E711
  E745  get flags again <E7FC>
  E748  has index buffer changed?
  E74A  no >>E751
  E74C  yes, checkpoint it <E09B>
  E74F  error? >>E711
  E751  ---
  E758  copy file identifier data to my variables (F300)
  E762  set DEVNUM (BF30)
  E765  BLKNUM = current DIR block (F01A)
  E76F  read DIR block <DDE1>
  E772  error? >>E711
  E774  copy directory header <D8A0>
  E777  are we in block with this file's entry? (F01C)
  E780  no >>E787
  E785  yes >>E78E
  E787  no, set new block number
  E78B  read it <DDE1>
  E78E  point at directory entry in block  <D692>
  E791  copy file entry from directory <D79D>
  E797  copy blocks_used count to entry (F318)
  E7A5  copy new EOF (F315)
  E7B0  and new key block no. (F30C)
  E7B9  isolate new storage type (F305)
  E7C3  combine it with name length (F01F)
  E7CB  and update type/len field in entry (F01F)
  E7CE  write entry back to directory <D6AB>
  E7D1  error? >>E7ED
  E7D6  turn off "write occured" flag (F31C)
  E7DE  same bitmap in memory (F019)
  E7E4  no, exit now >>E7EB
  E7E6  yes, checkpoint it also <DD86>
  E7EB   no errors, exit
  E7EC  RETURN
```

```
ProDOS MLI -- V1.0.1 -- 1 JAN 84                NEXT OBJECT ADDR: E7EC
-------------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------------


E7ED ********** CLOSE ERROR ****************************************
  E7ED  is this a close or flush all?
  E7F2  no >>E7FA
  E7F6  yes, save error code (F07E)
  E7F9  RETURN

  E7FA   else, real error right now
  E7FB  RETURN

E7FC ********** GET FCB FLAGS ***************************************

  E7FC  load FCB flags (F052)
  E7FF  from FCB (F308)
  E802   and exit

E803 ********** FILE ACCESS ERROR ***********************************

  E803  exit with file access error code
  E806  RETURN

E807 ************************************************************************
         ****** MLI SET_EOF CALL ******
         ******************************

  E807  get storage type <E40E>
  E80A  if DIR file...
  E80C  its an access error >>E803
  E80E  else, save type for truncate to
  E80F  mess with.
  E815  write access permitted? (F309)
  E81A  no, error >>E803
  E81C  check device status <E64E>
  E81F  error? >>E803
  E828  copy EOF from FCB (F315)
  E836  copy caller's new EOF
  E841  compare old EOF to new (F04A)
  E847  if less than or equal to... >>E84E
  E849  if greater... >>E863

         *** OLD EOF <= NEW EOF ***
         *** NO TRUNCATE NEEDED ***

  E84E  new eof beyond old
  E855  copy caller's EOF to FCB
  E860  exit by indicating flush needed >>EC50
```

```
ProDOS MLI -- V1.0.1 -- 1 JAN 84                NEXT OBJECT ADDR: E860
-------------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------------

         *** OLD EOF > NEW EOF ***
         *** TRUNCATE FILE     ***

  E863  flush first <E71C>
  E866  error? >>E806
  E868  $43/$49 --> end of data block I/O buffer
  E872  compare current mark to new EOF (F052)
  E87F  it is prior to EOF >>E898
  E887  if past EOF, force mark back to EOF (F052)
  E898  construct EOF block number and (F06A)
  E89B  byte offset into block from new (F086)
  E89E  EOF mark. (F06B)
  E8B6  on a block boundary? (F087)
  E8B9  yes >>E8D8
  E8BB  no, (F085)
  E8BF  decrement block by 1
  E8CD  but don't let it fall below 0
  E8D8  copy key block number (F052)
  E8E7  set blocks freed to zero
  E8EF  truncate file at new EOF <EC62>
  E8F2   save status
  E8FA  set new key block in FCB (F07F)
  E900  drop FCB block count by number (F318)
  E903  of blocks freed in truncate routine. (F082)
  E910  copy new storage type (F081)
  E91D  turn off all block allocation flags <DFA2>
  E920  update VCB free block count <EBDD>
  E92A  copy mark (F312)
  E932  force current mark to infinity (F312)
  E939  go set mark <DE3B>
  E93C  no errors? >>E945
  E93E   if error, indicate in saved status
  E944   but continue
  E945  copy caller's EOF to FCB <E84E>
  E94A   exit

E94B ************************************************************************
         ****** MLI GET_EOF CALL ******
         ******************************

  E94B  ---
  E950  copy EOF to caller's list (F315)
  E95C   exit -- no errors

E95D ************************************************************************
         ****** MLI NEW_LINE CALL ******
         ******************************
```

```
ProDOS MLI -- V1.0.1 -- 1 JAN 84                NEXT OBJECT ADDR: E95D
-------------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------------

  E95D  ---
  E95F  copy newline mask
  E968  and newline character
  E96E   return, no errors

E96F ************************************************************
         ****** MLI GET FILE INFO CALL ******
         ************************************

  E96F  get the file entry <D798>
  E972  ok? >>E9B6
  E974  no, bad path?
  E977  no, real error >>E9D3
  E979  else, make it VOL DIR type
  E97B  with name length = 0 (F01F)
  E980  no free blocks needed (F054)
  E986  go through the motions to update the (F051)
  E989  VCB block count. <DB71>
  E98F  copy blocks free from VCB (F215)
  E99B  copy total blocks on volume to AUX_ID (F213)
  E9A9  total - free = blocks_used (F054)
  E9B6  shift type down from high nibble (F01F)
  E9C2  copy the data to caller's parmlist (EFCC)
  E9D3   and exit

E9D4 ************************************************************
         ****** MLI SET FILE INFO CALL ******
         ************************************

  E9D4  get the file entry <D798>
  E9D7  error? >>E9FE
  E9D9  indicate backup needed now (BF95)
  E9E8  copy 13 parms from caller's list to (EFCC)
  E9EB  file entry staging area >>E9F2
  E9F2  ---
  E9F7  if any spurious access bits are on...
  E9FB  access error!
  E9FE  RETURN

  E9FF  else, anything in his modification date?
  EA03  no >>EA08
  EA05  yes, go update directory >>D6BB

  EA08  no, use system date then update directory >>D6AB
```

```
ProDOS MLI -- V1.0.1 -- 1 JAN 84                NEXT OBJECT ADDR: EA0B
-------------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------------

EA0B ************************************************************
         ****** MLI RENAME CALL ******
         ****************************

  EA0B  follow path to file <D7AB>
  EA0E  ok? >>EA4D
  EA10  no, bad name?
  EA12  no, real error >>EA2C

         *** RENAME VOLUME ***

  EA14  yes, copy new name  <EB35>
  EA17  error? >>EA2C
  EA19  get first length (F100)
  EA1D  get next (F100)
  EA20  bad path if more than one name for vol >>EAA1
  EA25  files open on volume? (F211)
  EA28  no, continue >>EA2E
  EA2A  yes, file open error
  EA2C  ---
  EA2D  RETURN

  EA2E  make type/len for a VOL DIR HDR
  EA35  write new name to VOL HDR <EB26>
  EA38  error? >>EAA3
  EA3F  copy new name to device's VCB (F100)
  EA4B   exit, no errors
  EA4C  RETURN

         *** RENAME FILE ***

  EA4D  get path index <EB43>
  EA50  copy old name with prefix to my buffer (F100)
  EA5C  copy new name to buffer <EB35>
  EA5F  error? >>EAA3
  EA61  get path index <EB43>
  EA67  compare all levels of names up to and (F600)
  EA6A   including the last. Find first which
  EA6B  differ.
  EA6F  save indicies into names which point to (F079)
  EA72  final name. (F07A)
  EA75  ---
  EA7F   exit if they match completely
  EA80  RETURN

  EA81  index to differing new name (F079)
  EA84  point past it (F100)
  EA8C  must be the last! (F100)
  EA8F  it isn't >>EAA1
  EA91  it is, (F07A)
  EA94  do the same with the old name (F600)
```

```
ProDOS MLI -- V1.0.1 -- 1 JAN 84                NEXT OBJECT ADDR: EA9F
-------------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------------

  EA9F  difference is only in last index? >>EAA5
  EAA1  no, bad path error
  EAA3  ---
  EAA4  RETURN

  EAA5  names good, follow path to new file <D7AB>
  EAA8  better get an error >>EAAE
  EAAA  if found, duplicate name in directory
  EAAD  RETURN

  EAAE  if error, better be file not found
  EAB0  or else its really an error... >>EAA3
  EAB2  copy old pathname again <D27F>
  EAB5  get its file entry <D798>
  EAB8  error? >>EAA3
  EABA  search FCB's <E1A9>
  EABF  exit if the file is open for write >>EAA3
  EAC4  does ACCESS permit rename?
  EAC6  yes >>EACC
  EAC8  no, access error
  EACA  ---
  EACB  RETURN

  EACC  get type/len from entry (F01F)
  EAD1  DIR file?
  EAD3  yes, ok >>EADD
  EAD5  seedling, sapling or tree?
  EAD7  yes, ok >>EADD
  EAD9  else, compatibility error
  EADD  copy new path again <EB35>
  EAE0  error? >>EAA3
  EAE2  get length of last name (F079)
  EAED  copy it and name to file entry buffer (F100)
  EAFD  combine new len with type (F100)
  EB03  DIR file?
  EB05  no, go update entry and exit >>EB23
  EB07  yes, (F030)
  EB11  read key block of this subdirectory <DDE1>
  EB14  error? >>EAA3
  EB19  copy new name to DIR HDR (F100)
  EB1E  and update directory's key block <EB26>
  EB21  error? >>EAA3
  EB23  go update directory entry and exit >>D6BB

EB26 ********* COPY PATH TO BUFF & WRITE *****************************

  EB26  copy type/len and path to my buffer
  EB32  go write the block >>DDDD
```

```
ProDOS MLI -- V1.0.1 -- 1 JAN 84                NEXT OBJECT ADDR: EB32
-------------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------------

EB35 ********* POINT TO NEW NAME ***************************************
          COPY TO BUFFER

  EB35  $48/$49 --> second pathname
  EB40  go copy it >>D28A

EB43 ********* LOAD PATH INDEX *****************************************

  EB43  load pathname index
  EB4A  (including prefix if any) (BF9A)
  EB4D  ---
  EB4F  RETURN

EB50 *************************************************************************
          ****** MLI DESTROY CALL ******
          ******************************

  EB50  get file entry <D798>
  EB53  error? >>EB9F
  EB55  find FCB if any <E1A9>
  EB58  FCB open? (F057)
  EB5B  no >>EB61
  EB5D  yes, file open error
  EB60  RETURN

  EB61  no free blocks needed
  EB69  go compute VCB free block count <DB66>
  EB6C  ok? >>EB73
  EB6E  error, disk full?
  EB71  no, real error >>EB9F
  EB73  DESTROY enabled in ACCESS? (F03D)
  EB78  yes >>EB7F
  EB7A  no, access error
  EB7F  check status of device (BF30)
  EB85  error? >>EB9F
  EB87  point to key block (F030)
  EB96  DIR file?
  EB9A  no >>EBA0
  EB9C  yes, handle differently >>EBF8

  EB9F  RETURN

          *** DESTROY NON-DIRECTORY FILE ***

  EBA0  set new storage type (F081)
  EBA7  zero EOF mark (F081)
  EBAD  byte offset = $200
  EBB2  free all blocks in file <EC62>
  EBB5  error? >>EB9F
  EBB7  free key block of seedling (F080)
```

```
ProDOS MLI -- V1.0.1 -- 1 JAN 84                NEXT OBJECT ADDR: EBC0
-------------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------------

  EBC0  error? >>EB9F
  EBC2  mark DIR entry free
  EBC7  decrement DIR file count (F013)
  EBD2  checkpoint volume bit map <DD86>
  EBD5  error? >>EB9F
  EBD7  update free block count in VCB <EBDD>
  EBDA  and go update the directory >>D6AB

          *** SUBROUTINE TO UPDATE FREE BLOCK ***
          *** COUNT IN VCB                    ***

  EBDD  add blocks freed to total free blocks (F051)
  EBE0  in VCB. (F082)
  EBF2  start next search for free blocks at
  EBF4  start of bitmap. (F21C)
  EBF7   exit

          *** DESTROY DIRECTORY FILE ***

  EBF8  DIR file?
  EBFA  no, error >>EC4B
  EBFC  read volume bitmap block <DD57>
  EBFF  error? >>EC4A
  EC01  BLKNUM = key block pointer (F030)
  EC0B  read it <DDE1>
  EC0E  errors? >>EC4A
  EC10  if DIR has any files... (F625)
  EC1A  access error
  EC1F  write back block marking entry free (F604)
  EC25  error? >>EC4A
  EC27  if "next_pointer" is zero.... (F602)
  EC31  go back and pretend it's a seedling >>EBB7
  EC33  else, (F603)
  EC36  free next block <DC27>
  EC39  error? >>EC4A
  EC3B  BLKNUM = next block (F602)
  EC45  read it <DDE1>
  EC48  if ok, continue in loop >>EC27
  EC4A   else, error exit

  EC4B  incompatible file format error

EC50 ********* SET WRITE OCCURED FLAG *********************************

  EC50   save some registers
  EC53  indicate write occured (F052)
  EC5E   restore regs and exit
  EC61  RETURN
```

```
ProDOS MLI -- V1.0.1 -- 1 JAN 84                NEXT OBJECT ADDR: EC62
-------------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------------

EC62 ********** TRUNCATE FILE AT EOF **********************************

  EC62  check storage type*16 (F081)
  EC65  seedling?
  EC67  yes >>EC74
  EC69  no, sapling?
  EC6B  yes >>EC77
  EC6D  no, tree?
  EC6F  yes >>EC7A
  EC71  no, die horribly <BF0C>
  EC74  go to seedling truncate >>ED46

  EC77  go to sapling truncate >>ED0D

  EC7A  truncate tree,
  EC7C  at most 128 blocks in master index (F088)
  EC7F  read the master index <ED71>
  EC82  error? >>ECDF
  EC84  at EOF yet? (F088)
  EC8A  yes >>ECE0

          *** FREE WHOLE INDEX BLOCKS AFTER EOF ***
          (free 8 subindex blocks each time the
          master index block is read since we must
          share its buffer)

  EC8C  copy up to 8 non-zero index block
  EC8E  numbers to (F600)
  EC91  a handy table (F08A)
  ECA2  ---
  ECAB  if there weren't 8 left to do, zero (F08A)
  ECAE  remainder of the table (F092)
  ECB4  ---
  ECB5  update master index counter (F088)
  ECBA  for all 8 entries: (F089)
  ECBD  set BLKNUM (F08A)
  ECC5  (exit when a 0 entry is found) >>EC7F
  ECCC  read the sub-index block <DDE1>
  ECCF  error?  >>ECDF
  ECD1  free all its blocks <EDA0>
  ECD4  error? >>ECDF
  ECDA  and loop to do all 8 >>ECBA
  ECDC  then go back and reread master index >>EC7F
  ECDE   normal exit
  ECDF  RETURN
```

```
ProDOS MLI -- V1.0.1 -- 1 JAN 84                NEXT OBJECT ADDR: ECDF
-------------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------------

  ECE0  now go free all the sub-index blocks (F084)
  ECE4  which follow EOF <EDA2>
  ECE7  error? >>ECDF
  ECE9  write back master index <DDDD>
  ECEC  error? >>ECDF
  ECEE  EOF in first subindex? (F084)
  ECF1  if so, demote to sapling file >>ED08
  ECF3  else, BLKNUM = subindex block which (F600)
  ECF6  contains the EOF mark
  ECFB  (exit if none there) >>ECDE
  ED02  else, read subindex block <DDE1>
  ED05  and continue below >>ED12
  ED07   unless there is an error

  ED08  demote tree to sapling <ED7E>
  ED0B  error? >>ECDF

          *** TRUNCATE SAPLING FILE ***

  ED0D  read key block <ED71>
  ED10  error? >>ECDF
  ED12  get LSB of block number (F085)
  ED16  if zero, no blocks to free >>ED22
  ED18  else, free rest of blocks in index <EDA2>
  ED1B  following the EOF. check for error >>ECDF
  ED1D  write index block back <DDDD>
  ED20  error? >>ECDF
  ED22  get LSB of block number (F085)
  ED25  might be block 0? >>ED3C
  ED27  no, get BLKNUM of data block (F600)
  ED2A  from index block
  ED2F  (no block allocated?) >>ECDE
  ED36  read data block <DDE1>
  ED39  and continue below >>ED4B
  ED3B   unless error occured

  ED3C  back to block 0? (F084)
  ED3F  no >>ED27
  ED41  yes, demote to seedling <ED7E>
  ED44  error? >>ED70

          *** TRUNCATE SEEDLING FILE ***

  ED46  read key block <ED71>
  ED49  error? >>ED70
  ED4B  first page? (F087)
  ED4E  yes >>ED56
  ED51  no, better be second >>ED6F
  ED53  get byte offset (F086)
  ED56  ---
```

```
ProDOS MLI -- V1.0.1 -- 1 JAN 84                NEXT OBJECT ADDR: ED58
-------------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------------

  ED58  zero beyond EOF mark (F700)
  ED66  in both pages if necessary (F600)
  ED6C  then write block back and exit >>DDDD

  ED6F   exit normally
  ED70  RETURN

ED71 ********** READ KEY BLOCK ***************************************

  ED71  BLKNUM = key block number (F07F)
  ED7B  exit by reading the block >>DDE1

ED7E ********** DEMOTE FILE TO SMALLER FILE TYPE**********************

  ED7E  free block (F080)
  ED87  error? >>ED9F
  ED89  get block from old index (F600)
  ED96  reduce storage type by one (F081)
  ED9E   and exit
  ED9F  RETURN

EDA0 ********** FREE ALL BLOCKS IN AN INDEX BLK **********************

  EDA0  ---
  EDA2  save BLKNUM
  EDA8  for each index entry after mark, (F05D)
  EDB3  if it is non-zero....
  EDBA  free the block <DC27>
  EDBD  error? >>EDCE
  EDBF  zero the index entry now (F05D)
  EDCA  ---
  EDCB  loop through all entries >>EDA8
  EDCE  ---
  EDD0  restore old BLKNUM
  EDD6   and exit

EDD7 ********** ALLOCATE I/O BUFFER **********************************

  EDD7  ---
  EDD9  get I/O buffer page number
  EDDC  can't be below $800
  EDDE  else, error >>EE22
  EDE0  can't be above $BC00
  EDE2  else, error >>EE22
  EDE7  $4A/$4B --> I/O buffer
  EDEB  must be page aligned! >>EE22
  EDF1  ---
  EDF2  check each page of I/O buffer for <EE5D>
  EDF5  prior allocation in system bit map (BF58)
  EE02  ---
  EE03  if ok, mark each page as allocated <EE5D>
```

```
ProDOS MLI -- V1.0.1 -- 1 JAN 84                NEXT OBJECT ADDR: EE06
-------------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------------

  EE06  in system memory bit map (BF58)
  EE13  assign buffer number (REFNUM*2) in FCB (F300)
  EE1B  and save buffer location in buffer list
  EE20   exit
  EE21  RETURN

  EE22  bad I/O buffer error
  EE25  RETURN

EE26 ********** LOCATE I/O BUFFER ***********************************

  EE26  ---
  EE27  AREG contains buffer number *2 (BF6E)
  EE2A  move buffer pointer to NXTBUF variable (F09D)
  EE33   exit

EE34 ********** FREE I/O BUFFER *************************************

  EE34  is buffer already free? <EE26>
  EE39  yes, exit >>EE5B
  EE3D  zero its address in system global page (BF6F)
  EE4A  ---
  EE4B  free each page in buffer <EE5D>
  EE4E  by marking system bit map
  EE5B   exit
  EE5C  RETURN

EE5D ********** LOCATE BIT MAP POSITION ******************************
          (GIVEN PAGE NUMBER)

  EE5D   XREG contains page number
  EE5E  compute page number times 8
  EE61  use as offset for bitmask (EFC0)
  EE68  page number / 8 = byte offset
  EE69   into bitmap
  EE6B   exit

EE6C ********** CHECK BUFFER VALIDITY ********************************
                  START > $200    END < $BF00

  EE6C  get buffer address (MSB)
  EE70  must be >$200 else error  >>EE22
  EE72  get length (F09B)
  EE78  compute last page no. of buffer
  EE7D  ---
  EE84  may not extend into $BF00
  EE86  else, error >>EE22
```

```
ProDOS MLI -- V1.0.1 -- 1 JAN 84                NEXT OBJECT ADDR: EE88
-------------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------------

          *** CHECK IF BLOCK OF MEMORY IS FREE ***

  EE89  ---
  EE8A  see if this page is allocated <EE5D>
  EE90  if so, error >>EE22
  EE92  else, check other page also
  EE96   then exit if both have been checked
  EE97  RETURN

EE98 ************************************************************
          ****** MLI GET_BUFF CALL ******
          ******************************

  EE98  get next available buffer
  EE9D  put its address in caller's parmlist
  EEA5   and exit
  EEA6  RETURN

EEA7 ************************************************************
          ****** MLI SET_BUFF CALL ******
          ******************************

  EEA7  mark his buffer allocated
  EEAC  error? >>EECE
  EEAE  get old buffer address (F09E)
  EEB8  free old buffer's pages in map <EE43>
  EEBF  copy old buffer contents
  EEC1  to new buffer
  EECD   then exit
  EECE  RETURN

EECF ********* GO TO QUIT CODE HANDLER *******************************

  EECF  enable 2nd 4K bank of language card (C083)
  EED2  (it lives at $D100-$D3FF) (C083)
  EED5  Save zeropage $00 through $03 on stack
  EEE1  Set ($00) -> $D100
  EEE3  Set ($02) -> $1000
  EEEF   Set Y = 0
  EEF0  3 pages of code to copy
  EEF2  ---
  EEF3  copy quit code handler to $1000
  EF01   Restore zero page to original state
  EF0D  enable 1st 4K bank of language card (C08B)
  EF10  (MLI) (C08B)
  EF15  point RESET vector at $1000 (03F2)
  EF1D  set power-up byte properly
  EF22  go to quit code handler at $1000 >>1000
```

```
ProOOS MLI -- Vl.Ø.1 -- 1 JAN 84                NEXT OBJECT AODR: EF22
-------------------------------------------------------------------
AODR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------

EF25 ***************************************************************
         *       OATA AREA                  *
         **********************************

EF25 ********** MLI COMMANO TABLE ***********************************
          IN HASH COOE ORDER: IF COMMAND IS...
          ABCO EFGH (IN BINARY BITS)
          INDEX IS COMPUTED AS:
             ØØØD EFGH
            +ØØØØ ABCO

  EF25  GET BUF
  EF26  UNUSED
  EF27  UNUSED
  EF28  UNUSED
  EF29  ALLOC INTERRUPT
  EF2A  OEALLOC INTERRUPT
  EF2B  UNUSEO
  EF2C  UNUSEO
  EF2D  REAO BLOCK
  EF2E  WRITE BLOCK
  EF2F  GET TIME
  EF3Ø  EXIT
  EF31  CREATE
  EF32  OESTROY
  EF33  RENAME
  EF34  SET FILE INFO
  EF35  GET FILE INFO
  EF36  ON LINE
  EF37  SET PREFIX
  EF38  GET PREFIX
  EF39  OPEN
  EF3A  NEWLINE
  EF3B  REAO
  EF3C  WRITE
  EF3D  CLOSE
  EF3E  FLUSH
  EF3F  SET MARK
  EF4Ø  GET MARK
  EF41  UNUSEO
  EF42  SET EOF
  EF43  GET EOF
  EF44  SET BUF

EF45 ********** PARAMETER COUNT TABLE *********************************
```

```
ProDOS MLI -- Vl.Ø.1 -- 1 JAN 84                NEXT OBJECT ADOR: EF45
-------------------------------------------------------------------
ADOR    OESCRIPTION/CONTENTS
-------------------------------------------------------------------

  EF45  GET BUF
  EF46  UNUSEO
  EF47  UNUSEO
  EF48  UNUSED
  EF49  ALLOC INTERRUPT
  EF4A  OEALLOC INTERRUPT
  EF4B  UNUSEO
  EF4C  UNUSEO
  EF4O  REAO BLOCK
  EF4E  WRITE BLOCK
  EF4F  GET TIME
  EF5Ø  EXIT
  EF51  CREATE
  EF52  OESTROY
  EF53  RENAME
  EF54  SET FILE INFO
  EF55  GET FILE INFO
  EF56  ON LINE
  EF57  SET PREFIX
  EF58  GET PREFIX
  EF59  OPEN
  EF5A  NEWLINE
  EF5B  REAO
  EF5C  WRITE
  EF5O  CLOSE
  EF5E  FLUSH
  EF5F  SET MARK
  EF6Ø  GET MARK
  EF61  UNUSEO
  EF62  SET EOF
  EF63  GET EOF
  EF64  SET BUF

EF65 ********** MLI COMMANO AOORESS TABLE ****************************

  EF65  CREATE
  EF67  OESTROY
  EF69  RENAME
  EF6B  SET FILE INFO
  EF6O  GET FILE INFO
  EF6F  ON LINE
  EF71  SET PREFIX
  EF73  GET PREFIX
  EF75  OPEN
  EF77  NEWLINE
  EF79  READ
  EF7B  WRITE
  EF7O  CLOSE
  EF7F  FLUSH
  EF81  SET MARK
```

```
ProDOS MLI -- V1.0.1 -- 1 JAN '84                NEXT OBJECT ADDR: EF83
------------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
------------------------------------------------------------------------

  EF83  GET MARK
  EF85  SET EOF
  EF87  GET EOF
  EF89  SET BUF
  EF8B  GET BUF

EF8D ********** MLI COMMAND INFO BYTE *********************************

       PATHNAME FLAG
         | REFERENCE NUMBER FLAG
         | | DATETIME STAMP FLAG
         | | |     COMMAND NUMBER
         | | |      |
  EF8D   1 0 1  - 00
  EF8E   1 0 1  - 01
  EF8F   1 0 1  - 02
  EF90   1 0 1  - 03
  EF91   1 0 0  - 04
  EF92   0 0 0  - 05
  EF93   0 0 0  - 06
  EF94   0 0 0  - 07
  EF95   1 0 0  - 08
  EF96   0 1 0  - 09
  EF97   0 1 0  - 0A
  EF98   0 1 0  - 0B
  EF99   0 0 1  - 0C
  EF9A   0 0 1  - 0D
  EF9B   0 1 0  - 0E
  EF9C   0 1 0  - 0F
  EF9D   0 1 0  - 10
  EF9E   0 1 0  - 11
  EF9F   0 1 0  - 12
  EFA0   0 1 0  - 13

EFA1 ********** CONSTANTS - DATA AREA *********************************

  EFA1  Blocks Used
  EFA3  End of File
  EFA6  Special ID (Must be 5 bits on)
  EFA7  'HUSTON!'          Author's name
  EFAE  Previous Block of Vol Dir Key Block
          ---
        THE FOLLOWING IS COPIED TO SUBDIR HDR+$20
  EFB0  Version of ProDOS
  EFB1  Minimum Version
  EFB2  Access Byte  (D|Rn|B|000|W|R)
  EFB3  Entry Length
  EFB4  Entries per Block
  EFB5  File Count
  EFB7  Parent LSB (copied to SUBDIR HDR +$20)
          ---
```

```
ProDOS MLI -- V1.0.1 -- 1 JAN 84                 NEXT OBJECT ADDR: EFB8
------------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
------------------------------------------------------------------------

  EFB8  File Type  (Directory)
  EFB9  Block Number
  EFBB  Number of Blocks
  EFBD  End of File

EFC0 ********** BITMASK TABLE ***************************************

  EFC0  10000000
  EFC1  01000000
  EFC2  00100000
  EFC3  00010000
  EFC4  00001000
  EFC5  00000100
  EFC6  00000010
  EFC7  00000001

EFC8 ********** OFFSETS TO DATA AT $F300 ******************************

  EFC8  Key Block

  EFCA  # Blocks Used

  EFCC  End of File

EFCF ********** SET/GET FILE_INFO OFFSETS ******************************

  EFCF  Access
  EFD0  File Type
  EFD1  Aux Type
  EFD3  Storage Type
  EFD4  Blocks Used (MSB on means GET only no SET)

  EFD6  Datetime (Last Mod)

  EFDA  Datetime (Creation)

EFDE ********** FATAL ERROR MESSAGE **********************************

  EFDE  '             INSERT SYSTEM DISK AND RESTART

  F006  ---

F006 ********** VARIABLES - DATA AREA *********************************

  F006  Parent Pointer Block
  F008  Parent Entry Number
  F009  Parent Entry Length
  F00A  Datetime (Creation)
  F00E  Version
  F00F  Min Version
  F010  Access Byte
```

```
ProDOS MLI -- V1.0.1 -- 1 JAN 84                NEXT OBJECT ADDR: F011
-------------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------------

  F011  Entry Length
  F012  Entries per Block
  F013  File Count
  F015  Bit Map Pointer
  F017  Total Blocks
          THE FOLLOWING 6 BYTES UNIQUELY IDENTIFY
          A FILE:
  F019  Device Number
  F01A  Current Directory Block Number (HDR)
  F01C  Block Number of File Entry in Directory
  F01E  File Entry Number in Directory

F01F ********** FILE ENTRY BUFFER ************************************

  F01F  Type/Length  (TTTTLLLL)
  F020  File Name (Max 15) >>000F
  F02F  File Type
  F030  Key Pointer
  F032  Blocks Used
  F034  End of File
  F037  Datetime (Creation)
  F03B  Version
  F03C  Min Version
  F03D  Access Attribute
  F03E  Aux Type (Load Address/Record Length)
  F040  Datetime (Last Mod)

  F044  Header Pointer

F046 ********** Variable Work Area ************************************

  F046  3 Byte Scratch

  F049  ---

  F04A  End of File

  F04D  Previous Mark

  F050  Compare Vol Name Scratch
  F051  Offset into VCB Table ($F200)
  F052  Offset into FCB Table ($F300)
  F053  Free FCB found Flag

  F054  Number of Free Blocks needed

  F056  Storage Type
          Number of Entries Examined or..
  F057  FCB already open flag
  F058  File Count
```

```
ProDOS MLI -- V1.0.1 -- 1 JAN 84                NEXT OBJECT ADDR: F058
-------------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------------

  F05A  Entries/Block Loop Count/Free FCB's refnum
          Free Entry Found Flag (if > 0) or..
          # of 1st bitmap block with free bit on or..
  F05B  bit for free
  F05C  # Blocks in Bitmap left to search
  F05D  Y Register temp
  F05E  Pathname Length
  F05F  Devnum for Prefix Directory Header
  F060  Block of Prefix Directory Header
  F062  Bitmap Byte Offset in Page
  F063  Bitmap Page Offset
  F064  Bitmap Buffer Page (0 or 1)
  F065  Bitmap Flag (if $80, needs writing)
  F066  Bitmap DEVNUM
  F067  Bitmap Block Number
  F069  Bitmap Block offset for Multiblock Bitmaps

          New Mark to be Positioned to for Set Mark
          or New Moving Mark (for READ)
  F06A  or New EOF for SET_EOF

  F06D  Request Count (Read/Write etc.)
  F06F  Multi-Block I/O count
  F070  Newline character
  F071  Newline mask
  F072  I/O Transfer occurred flag
  F073  MLI Command * 2
  F074  ORed into Access Flags ($20 - Backup)
  F075  Duplicate Volume Flag (if $FF)
  F076  Duplicate Volume's VCB index
  F077  MLI function code (low 5 bits)
          Characters in current Pathname indx lvl or
  F078  ONLINE: volname len - loop index
  F079  new pathname: index to last name
          old pathname: index to last name or..
  F07A  ONLINE: index to data buffer
  F07B  Old PFIXPTR value
  F07C  Pathname fully qualified flag (if $FF)
          Pathname: temp save area for index or..
  F07D  ONLINE: DEVCNT
  F07E  close-all error code
  F07F  Set EOF: new Key Block pointer
  F081  New storage type (SET_EOF)
  F082  Freed Blocks count
  F084  EOF Block number (MSB then LSB)
  F086  EOF byte offset into Block
  F088  EOF - Master index counter
  F089  Save area for index into table below
```

```
ProDOS MLI -- V1.0.1 -- 1 JAN 84                NEXT OBJECT ADDR: F089
-------------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------------

F08A ********** DEVICE TABLE BUILT BY ONLINE ************************
         (also used by SET_EOF to keep track of
         8 blocks to be freed at a time)

  F08A  device table part one
  F092  device table part two

  F09A  length of path, etc.
  F09D  next buffer address
  F09F  6 byte zeropage savearea >>0006
  F0A5  not used >>000A
  F0AF  16 byte interrupt savearea >>0006
  F0B5  Jump Vector

  F0B7   not used >>0049

F100 ********** PATHNAME - DATA AREA *********************************
           ---------------------------------
           | L1 | NAME1 | L2 | NAME2 | .. | 00
           ---------------------------------
         Prefix is at top of buffer such that a
         negative index may be used to use it,
         wrapping around to the pathname again.

  F100  pathname buffer >>0100

F200 ********** VOLUME CONTROL BLOCKS ********************************

         VCB0 starts here..
  F200  Length  (0000LLLL)
  F201  File Name (Max 15) >>000F
  F210  Unit Number
  F211  Files Open Flag (if $FF)
  F212  Total Blocks
  F214  Blocks Free
  F216  Block Number of Vol Dir Key Block
  F218  not used
  F219  not used
  F21A  Bit Map Pointer
          Block offset into multi-block bitmap of
  F21C  next free bit.
  F21E  Count of open files

  F220  VCB1 through VCB7 >>00E0
```

```
ProDOS MLI -- V1.0.1 -- 1 JAN 84                NEXT OBJECT ADDR: F300
-------------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------------

F300 ********* FILE CONTROL BLOCKS ***********************************

         FCB0 starts here..
  F300  Reference Number

         THE FOLLOWING 6 BYTES ARE THE FILE ID
  F301  Device Number
  F302  Dir Block HDR for Dir describing this File
  F304  Dir Block containing entry itself
  F306  File entry # in this Directory

  F307  Storage Type
          Flags
          1XXX XXXX   Index Block Buffer Changed
          X1XX XXXX   Data Block Buffer Changed
          XX1X XXXX   Unused
          XXX1 XXXX   Directory entry needs update
          XXXX 1XXX   Storage Type Changed
          XXXX X1XX   Allocate new Master Index Block
          XXXX XX1X   Allocate new Sub Index Block
  F308  XXXX XXX1   Allocate new Data Block
  F309  Access Byte
  F30A  Newline Character
  F30B  Buffer Number  (REF_NUM * 2)
  F30C  Master Index/Key Block Number
  F30E  Current Index Block
  F310  Current Data Block
  F312  Mark
  F315  End of File
  F318  Blocks Used
  F31A  not used
  F31B  Level
  F31C  Flag - Write occured if MSB on
  F31D  not used
  F31F  Newline Enable Mask

  F320  FCB1 through FCB7 >>00E0

F400 ********** BITMAP BUFFER ****************************************

  F400  Buffer 1st half >>0100

  F500  Buffer 2nd half >>0100

F600 ********** PRIMARY BUFFER ***************************************
         (used for several things, VOL DIR HDR is
         mapped into it below)
```

```
ProDOS MLI -- V1.0.1 -- 1 JAN 84                NEXT OBJECT ADDR: F600
-------------------------------------------------------------------------
ADDR     DESCRIPTION/CONTENTS
-------------------------------------------------------------------------


  F600  Pointer Fields
      .   *** VOLUME DIRECTORY HEADER ***
  F604  Type/Length  (TTTTLLLL)
  F605  File Name  (Max 15) >>000F
  F614  Reserved >>0008
  F61C  Creation Datetime
  F620  Version
  F621  Min Version
  F622  Access Byte
  F623  Entry Length
  F624  Entries per Block
  F625  File Count
  F627  Bitmap Pointer
  F629  Total Blocks
  F62B  (remainder of first page of block) >>00D5
  F700  (second page of block) >>0100
```

ProDOS SYSTEM GLOBAL PAGE--MLI Global Page

Portions of this page of memory are rigidly defined by the MLI and are unlikely to move in later versions of ProDOS. However, some portions are less stable and could change in future releases.

ProDOS System Global Page NEXT OBJECT ADDRESS: BF00

ADDR	LABEL	CONTENTS
		Jump Vectors
BF00-BF02	ENTRY	JMP to MLI.
BF03-BF05	JSPARE	JMP to system death code (via $BFF6).
BF06-BF08	DATETIME	JMP to Date/Time routine (RTS if no clock).
BF09-BF0B	SYSERR	JMP to system error handler.
BF0C-BF0E	SYSDEATH	JMP to system death handler.
BF0F	SERR	System error number.
		Device Information
BF10-BF11	DEVADR01	Slot 0 reserved
BF12-BF13	DEVADR11	Slot 1, drive 1 device driver address.
BF14-BF15	DEVADR21	Slot 2, drive 1 device driver address.
BF16-BF17	DEVADR31	Slot 3, drive 1 device driver address.
BF18-BF19	DEVADR41	Slot 4, drive 1 device driver address.
BF1A-BF1B	DEVADR51	Slot 5, drive 1 device driver address.
BF1C-BF1D	DEVADR61	Slot 6, drive 1 device driver address.
BF1E-BF1F	DEVADR71	Slot 7, drive 1 device driver address.
BF20-BF21	DEVADR02	Slot 0 reserved.
BF22-BF23	DEVADR12	Slot 1, drive 2 device driver address.
BF24-BF25	DEVADR22	Slot 2, drive 2 device driver address.
BF26-BF27	DEVADR32	/RAM device driver address (need extra 64K).
BF28-BF29	DEVADR42	Slot 4, drive 2 device driver address.
BF2A-BF2B	DEVADR52	Slot 5, drive 2 device driver address.
BF2C-BF2D	DEVADR62	Slot 6, drive 2 device driver address.
BF2E-BF2F	DEVADR72	Slot 7, drive 2 device driver address.
BF30	DEVNUM	Slot and drive (DSSS0000) of last device.
BF31	DEVCNT	Count (minus 1) of active devices.
BF32-BF3F	DEVLST	List of active devices (slot, drive and identification--DSSSIIII).
BF40-BF4F		Copyright notice.
BF50-BF55	IRQXITX	Switch in language card and call IRQ handler at $FFD8.
BF56-BF57	TEMP	Temporary storage for IRQ code.
BF58-BF6F	BITMAP	Bitmap of low 48K of memory.
BF70-BF71	BUFFER1	Open file 1 buffer address.
BF72-BF73	BUFFER2	Open file 2 buffer address.
BF74-BF75	BUFFER3	Open file 3 buffer address.
BF76-BF77	BUFFER4	Open file 4 buffer address.
BF78-BF79	BUFFER5	Open file 5 buffer address.
BF7A-BF7B	BUFFER6	Open file 6 buffer address.
BF7C-BF7D	BUFFER7	Open file 7 buffer address.
BF7E-BF7F	BUFFER8	Open file 8 buffer address.

ProDOS System Global Page NEXT OBJECT ADDRESS: BF80

ADDR	LABEL	CONTENTS
		Interrupt Information
BF80-BF81	INTRUPT1	Interrupt handler address (highest priority).
BF82-BF83	INTRUPT2	Interrupt handler address.
BF84-BF85	INTRUPT3	Interrupt handler address.
BF86-BF87	INTRUPT4	Interrupt handler address (lowest priority).
BF88	INTAREG	A-register savearea.
BF89	INTXREG	X-register savearea.
BF8A	INTYREG	Y-register savearea.
BF8B	INTSREG	S-register savearea.
BF8C	INTPREG	P-register savearea.
BF8D	INTBANKID	Bank ID byte (ROM, RAM1, or RAM2).
BF8E-BF8F	INTADDR	Interrupt return address.
		General System Info
BF90-BF91	DATE	YYYYYYYM MMMDDDDD.
BF92-BF93	TIME	...HHHHH ..MMMMMM.
BF94	LEVEL	Current file level.
BF95	BUBIT	Backup bit.
BF96-BF97	SPARE1	Currently unused.
BF98	MACHID	Machine ID byte. 00.. 0... II 01.. 0... II+ 10.. 0... IIe 11.. 0... III emulation 00.. 1... Future expansion 01.. 1... Future expansion 10.. 1... IIc 11.. 1... Future expansion ..00 .... Unused ..01 .... 48K ..10 .... 64K ..11 .... 128K .... .X.. Reserved .... ..0. No 80-column card .... ..1. 80-column card present ....0 No compatible clock ....1 Compatible clock present
BF99	SLTBYT	Slot ROM map (bit on indicates ROM present).
BF9A	PFIXPTR	Prefix flag (0 indicates no active prefix).
BF9B	MLIACTV	MLI active flag (1... .... indicates active).
BF9C-BF9D	CMDADR	Last MLI call return address.
BF9E	SAVEX	X-register savearea for MLI calls.
BF9F	SAVEY	Y-register savearea for MLI calls.

ProDOS System Global Page NEXT OBJECT ADDRESS: BFA0

ADDR	LABEL	CONTENTS
		Language Card Bank Switching Routines
BFA0-BFCF		Language card entry and exit routines.
BFA0	EXIT	
BFAA	EXIT1	
BFB5	EXIT2	
BFB7	MLIENT1	
		Interrupt Routines
BFD0-BFF3		Interrupt entry and exit routines.
BFD0	IRQXIT	
BFDF	IRQXIT1	
BFE2	IRQXIT2	
BFE7	ROMXIT	
BFEB	IRQENT	
		Data
BFF4	BNKBYT1	Storage for byte at $E000.
BFF5	BNKBYT2	Storage for byte at $D000.
BFF6-BFFB		Switch on language card and call system death handler ($D1E4).
		Version Information
BFFC	IBAKVER	Minimum version of Kernel needed for this interpreter.
BFFD	IVERSION	Version number of this interpreter.
BFFE	KBAKVER	Minimum version of Kernel compatible with this Kernel.
BFFF	KVERSION	Version number of this Kernel.

```
ProDOS Quit Code -- V1.0.1 -- 1 JAN 84          NEXT OBJECT ADDR: 1000
-------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------

1000    MODULE STARTING ADDRESS

            ***********************************
            *                                 *
            * Quit Code:                      *
            *     Resides in alternate 4K     *
            *     bank ($D000) and is         *
            *     executed at $1000           *
            *                                 *
            *                                 *
            * Versions 1.0.1 -- 1 JAN 84      *
            *                                 *
            ***********************************

1000 ********** ZERO PAGE EQUATES ************************************

0024    Cursor Horizontal
0025    Cursor Vertical

1000 ********** EXTERNAL EQUATES *************************************

0280    Prefix Buffer
1800    Buffer
2000    Buffer
BF00    MLI Entry
BF58    Bitmap

1000 ********** SOFT SWITCHES ****************************************

C000    Keyboard
C000    Disable 80 column store
C00C    Disable 80 column card
C00F    Select alternate character set
C010    Keyboard Strobe
C082    ROM select

1000 ********** MONITOR EQUATES **************************************

FC58    Home
FC9C    Clear to end of line
FD0C    Read a key
FD8E    Output a Carriage Return
FDED    Output a Character
FE89    Set Keyboard
FE93    Set Video
FF3A    Sound Bell
```

```
ProDOS Quit Code -- V1.0.1 -- 1 JAN 84          NEXT OBJECT ADDR: 1000
-------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------

1000 ********** INITIALIZATION ****************************************

  1000  Select ROM (C082)
  1003  Set Video <FE93>
  1006  Set Keyboard <FE89>
  1009  Disable 80 column card (C00C)
  100C  Select Alternate character set (C00F)
  100F  Disable 80 column store  (C000)

1012 ********** INITIALIZE MEMORY BITMAP ******************************

  1012  Mark pages $0, $1, $4 through $7
  1014  and $BF as in use

1027 ********** DISPLAY CURRENT PREFIX ********************************

  1027  Clear Screen and Home cursor <FC58>
  102A  Go down 1 line <FD8E>
  102D  Get Pointer to Prompt1 (Prefix)
  102F  and store it in Print Routine (11E9)
  1037  Call Print Routine <11E6>
  103A  Position to line 3
  1041  Call MLI (GET_PREFIX) <BF00>
  1044  Data: GET_PREFIX command number
  1045  Data: Pointer to Parameter list
  1047  Terminate Prefix with 0 (0280)
  104A  for Print routine
  104F  Get Pointer to Prefix
  1051  and store it in Print Routine (11E9)
  1059  And Print it <11E6>

105C ********** GET PREFIX NAME ***************************************

  105C  Initialize counter
  1063  Read a key <FD0C>
  1066  Is it CARRIAGE RETURN?
  1068  Yes, then accept Prefix >>10B8
  106A  No, then save character
  106B  Clear to end of line <FC9C>
  106E  Retrieve character
  106F  Is it ESCAPE?
  1071  Yes, then start all over again >>1027
  1073  Is it CANCEL?
  1075  Yes, then start all over again >>1027
  1077  Is it TAB?
  1079  Yes, then sound Bell, get another character >>108E
  107B  Is it BACKSPACE?
  107D  No, then keep checking >>108C
  107F  Yes, then is there room to move back?
  1081  No, then don't try >>1086
```

```
ProDOS Quit Code -- V1.0.1 -- 1 JAN 84          NEXT OBJECT ADDR: 1083
-------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------

  1083  Decrement cursor horizontal position
  1085  Decrement counter
  1086  Clear to end of line <FC9C>
  1089  Try again >>1063

  108C  Continue if greater than or equal to BACKSPACE >>1094
  108E  Else, sound Bell <FF3A>
  1091  Try again >>1063

  1094  Is it less than or equal to "Z"?
  1096  Yes, keep checking >>109A
  1098  Turn off lowercase
  109A  Is it less than "."?
  109C  Yes, Invalid - try again >>108E
  109E  Is it greater than "Z"?
  10A0  Yes, Invalid - try again >>108E
  10A2  Is it less than or equal to "9"?
  10A4  Yes, keep checking >>10AA
  10A6  Is it less than "A"?
  10A8  Yes, Invalid - try again >>108E
  10AA  Else, valid character - increment counter
  10AB  Found 39 characters?
  10AD  Yes, then start all over >>1075
  10AF  Put valid character in buffer (0280)
  10B2  and Print it <FDED>
  10B5  Go back for more >>1063

  10B8  Check counter
  10BA  If 0 then go on >>10CE
  10BC  Else, save length (0280)
  10BF  Call MLI (SET_PREFIX) <BF00>
  10C2  Data: SET_PREFIX command number
  10C3  Data: Pointer to Parameter list
  10C5  Carry on if no error >>10CE
  10C7  Sound Bell <FF3A>
  10CA  Force branch to
  10CC  always be taken >>1075

10CE ********** GET APPLICATION NAME *********************************

  10CE  Clear Screen and Home cursor <FC58>
  10D1  Go down 1 line <FD8E>
  10D4  Get Pointer to Prompt2 (Application)
  10D6  and store it in Print Routine (11E9)
  10DE  Print it <11E6>
  10E1  Position to line 3
  10E8  Initialize counter
  10EA  Output a RUB
  10F1  Poll Keyboard latch (C000)
  10F4  Loop until keypress found >>10F1
  10F6  Clear latch (C010)
```

```
ProDOS Quit Code -- V1.0.1 -- 1 JAN 84          NEXT OBJECT ADDR: 10F9
-------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------

  10F9  Is it ESCAPE?
  10FB  No, keep checking >>1103
  10FD  Yes, get Cursor horizontal position
  10FF  If not 0 try again  >>10CE
  1101  If 0 start all over again >>10CC
  1103  Is it CANCEL?
  1105  Yes, try again >>10CE
  1107  Is it TAB?
  1109  Yes, sound Bell - try again >>1114
  110B  Is it BACKSPACE?
  110D  No, keep checking >>1112
  110F  Yes, then handle it >>11D0

  1112  Continue if greater than or equal to BACKSPACE >>111A
  1114  Sound Bell <FF3A>
  1117  Go back and try again >>10EA

  111A  Is it CARRIAGE RETURN?
  111C  Yes, then go load Application >>1147
  111E  Is it less than or equal to "Z"?
  1120  Yes, keep checking >>1124
  1122  Turn off lower case
  1124  Is it less than "."?
  1126  Yes, Invalid - try again >>1114
  1128  Is it greater than "Z"?
  112A  Yes, Invalid - try again >>1114
  112C  Is it less than or equal to "9"?
  112E  Yes, keep checking >>1134
  1130  Is it less than "A"?
  1132  Yes, Invalid - try again >>1114
  1134  Else, valid character - save it
  1135  Clear to end of line <FC9C>
  1138  Retrieve character
  1139  and Print it <FDED>
  113C  Increment counter
  113D  Found 39 characters?
  113F  Yes, start again >>1105
  1141  No, save character in buffer (0280)
  1144  and go get another >>10EA

1147 ********** LOAD AND EXECUTE APPLICATION ************************

  1147  Output a blank
  114C  Store length of Application name (0280)
  114F  Call MLI (GET_FILE_INFO) <BF00>
  1152  Data: GET_FILE_INFO command number
  1153  Data: Pointer to Parameter list
  1155  Continue if no error >>115A
  1157  Else, go to Error Handler >>11F6
```

```
ProDOS Quit Code -- V1.0.1 -- 1 JAN 84          NEXT OBJECT ADDR: 1157
-------------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------------

  115A  Get File Type (12D5)
  115D  Is it ProDOS System file?
  115F  Yes, continue >>1166
  1161  No, indicate Error $01
  1163  Go to Error Handler >>11F6

  1166  Set Reference number to 0
  116B  Call MLI (CLOSE) <BF00>
  116E  Data: CLOSE command number
  116F  Data: Pointer to Parameter list
  1171  Continue if no error >>1176
  1173  Else, go to Error Handler >>11F6
  1176  Get Access Byte (12D4)
  117B  Yes, >>1182
  117D  No, Indicate Error $27
  117F  Go to Error Handler >>11F6

  1182  Call MLI (OPEN) <BF00>
  1185  Data: OPEN command number
  1186  Data: Pointer to Parameter list
  1188  Continue if no error >>118D
  118A  Else, go to Error Handler >>11F6

  118D  Get Reference Number (12E8)
  1190  and update READ and (12EC)
  1193  GET_EOF parameter lists (12F4)
  1196  Call MLI (GET_EOF) <BF00>
  1199  Data: GET_EOF command number
  119A  Data: Pointer to Parameter list
  119C  Continue if no error >>11A1
  119E  Else, go to Error Handler >>11F6

  11A1  Is EOF mark less than $10000 (12F7)
  11A4  Yes, continue on >>11AB
  11A6  No, Indicate Error $27
  11A8  Go to Error Handler >>11F6

  11AB  Transfer EOF to Request count (12F5)
  11AE  in READ parameter list (12EF)
  11B7  Call MLI (READ) <BF00>
  11BA  Data: READ command number
  11BB  Data: Pointer to Parameter list
  11BD  Save status of READ
  11BE  Call MLI (CLOSE) <BF00>
  11C1  Data: Get_Prefix command number
  11C2  Data: Pointer to Parameter list
  11C4  Continue if no error >>11CA
  11C6  Else, retrieve status
  11C7  and go to Error Handler >>11F6
```

```
ProDOS Quit Code -- V1.0.1 -- 1 JAN 84          NEXT OBJECT ADDR: 11C7
-------------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------------

  11CA  Was READ good?
  11CB  No, go to Error Handler >>11C7
  11CD  Yes, execute application >>2000

11D0 ********** BACKSPACE ROUTINE ************************************

  11D0  Get cursor position horizontal
  11D2  If 0 exit routine >>11E3
  11D4  Decrement counter
  11D5  Output a space
  11DA  Move cursor back 2 spaces
  11DE  Output a space <FDED>
  11E1  Move cursor back 1 space
  11E3  Return to get another character >>10EA

11E6 ********** PRINT TEXT ROUTINE ***********************************

  11E6  Initialize offset
  11E8  Get a character (11E8)
  11EB  If it is 0 then exit >>11F5
  11EF  Output it <FDED>
  11F2  Increment offset
  11F3  Get another character unless we've done 256 >>11E8
  11F5  Return to caller

11F6 ********** PRINT ERROR MESSAGE **********************************

  11F6  Save Accumulator (Error Number)
  11F8  Position to line 12
  11FF  Get Error number
  1201  Is it $01?
  1203  No, then keep checking >>1211
  1205  Get Pointer to Error1 (Not System file)
  1207  and store it in Print Routine (11E9)
  120F  Branch always taken >>1237
  1211  Is it $40?
  1213  Yes, then indicate Error3 >>122D
  1215  Is it $44?
  1217  Yes, then indicate Error3 >>122D
  1219  Is it $45?
  121B  Yes, then indicate Error3 >>122D
  121D  Is it $46?
  121F  Yes, then indicate Error3 >>122D
  1221  Else, Get Pointer to Error2 (I/O Error)
  1223  and store it in Print Routine (11E9)
  122B  Branch always taken >>1237
  122D  Get Pointer to Error3 (Path not found)
  122F  and store it in Print Routine (11E9)
  1237  Print Error message <11E6>
  123A  Position to line 0
```

```
ProDOS Quit Code -- V1.0.1 -- 1 JAN 84          NEXT DBJECT ADDR: 123E
-------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------

  123E  Return to Get Application code >>10D1

1241 ********** ASCII TEXT ****************************************

        Prompt1
  1241  'ENTER         PREFIX (PRESS "RETURN" TD ACCEPT)'

        Prompt2
  1269  'ENTER         PATHNAME DF NEXT APPLICATION'

        Error1
  128C  Ring Bell
  128D  'NDT          A TYPE "SYS" FILE'

        Error2
  12A3  Ring Bell
  12A4  'I/D          ERROR            '

        Error3
  12BA  Ring Bell
  12BB  'FILE/PATH    NDT FOUND  '

12D1 ********** PARAMETER LISTS ***********************************

        GET_FILE_INFO Parmlist
  12D1  Parmcount
  12D2  Pathname
  12D4  Access
  12D5  File Type
  12D6  Aux Type
  12D8  Storage Type
  12D9  Blocks Used
  12DB  Datetime (modified)
  12DF  Datetime (creation)

        OPEN Parmlist

  12E3  Parmcount
  12E4  Pathname
  12E6  I/O Buffer
  12E8  Reference Number

        CLOSE Parmlist

  12E9  Parmcount
  12EA  Reference Number
```

```
ProDOS Quit Code -- V1.0.1 -- 1 JAN 84          NEXT OBJECT ADDR: 12EA
-------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------

        READ Parmlist

  12EB  Parmcount
  12EC  Reference Number
  12ED  Data Buffer
  12EF  Request Count
  12F1  Transfer Count

        GET_EDF Parmlist

  12F3  Parmcount
  12F4  Reference Number
  12F5  EOF Mark

        GET/SET_PREFIX Parmlist

  12F8  Parmcount
  12F9  Pathname

          ******* UNUSED ********************

  12FB  --- >>0005
```

```
Disk II Device Driver -- V1.0.1 -- 1 JAN 84     NEXT OBJECT ADDR: F800
-------------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------------

F800    MODULE STARTING ADDRESS

            ***********************************
            *                                 *
            *  Disk Device Driver:            *
            *     Resides from $F800 - $FEFF  *
            *                                 *
            *                                 *
            *                                 *
            *                                 *
            *                                 *
            *                                 *
            *  Version 1.0.1 -- 1 JAN 84      *
            *                                 *
            ***********************************

F800 ********** ZERO PAGE EQUATES ************************************

003A    Checksum
003A    Workbyte
003E    Slot (Temporary)
0042    Command
0043    Unit Number
0044    I/O Buffer Pointer (low)
0045    I/O Buffer Pointer (high)
0046    Block Number (low)
0047    Block Number (high)

F800 ********** INTERNAL EQUATES *************************************

1000    Dummy Block Buffer (1st half)
1100    Dummy Block Buffer (2nd half)

F800 ********** EXTERNAL EQUATES *************************************

C080    Phase Zero Off
C088    Motor Off
C089    Motor On
C08A    Drive Select
C08C    Read Data Register
C08D    Write Data Register
C08E    Set Read Mode
C08F    Set Write Mode
C0EC    Read Data Register (slot 6)

F800 ********** CONVERT BLOCK TO TRACK/SECTOR ************************
```

```
Disk II Device Driver -- V1.0.1 -- 1 JAN 84     NEXT OBJECT ADDR: F800
-------------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------------

 F800  Get Block Number
 F804  Is Block Number good? (FB56)
 F807  Yes, if less than $100 >>F810
 F80A  No, if greater than or equal to $200  >>F836
 F80E  No, if greater than or equal to $118 >>F836
 F810  Convert Block Number to a Track and Sector
 F812  ---
 F816  0000000T TTTTTABC
 F817        .      . >>F812
 F819      .      .
 F81A    .        . >>F81E
 F81C  00TTTTTT 0000BC0A
 F81E  ---
 F822   Preserve Sector Number
 F823  Execute command <F83A>
 F826  Restore Sector Number - Was prior action ok?
 F827  No, then exit >>F832
 F829  Increment Buffer Pointer
 F82B  Increment Sector Number by 2 for rest of Block
 F82D  Execute command <F83A>
 F830  Decrement Buffer Pointer (to start of block)
 F832  Get error number (if any - 0 indicates no error) (FB58)
 F835  Return to caller

F836 ********** I/O ERROR ROUTINE ************************************

 F836  Indicate "I/O Error"
 F838  Set Carry flag
 F839  Return to caller

F83A ********** MAIN ROUTINE ****************************************

 F83A  Set recalibration count to 1
 F83F  Preserve sector number (FB57)
 F842  Get "Unitnum"   DSSS0000
 F844  Strip out Drive 0SSS0000
 F846  Preserve slot number
 F848  Check for slot change, turn off motor if so <FE9B>
 F84B  See if motor is on <FCDA>
 F84E  Save test results
 F851  Initialize counter for delay routine (FB70)
 F856  See if slot or drive has changed (FB59)
 F859  Update "current" unit number (FB59)
 F85C  Save test results
 F85D  Put drive number in Carry flag
 F85E  Turn motor on (C089)
 F864  Select appropriate drive (C08A)
 F867  Check test results - Same slot/drive?
 F868  Yes, then skip delay >>F874
 F86B  Wait for new Drive
```

```
Disk II Device Driver -- V1.0.1 -- 1 JAN 84     NEXT OBJECT ADDR: F86D
-------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------

  F86D  to come up to speed <FB85>
  F874  Is command a status request?
  F876  Yes, then do not move disk arm >>F87E
  F878  Get track number for current request (FB56)
  F87B  And go there <F90C>
  F87E  Check test results - Was motor on?
  F87F  Yes, then skip delay >>F890
  F881  Wait for Drive
  F883  to come up to speed <FB85>
  F88B  Is motor on yet? <FCDA>
  F88E  No, then exit with error >>F8EC
  F890  Is command a "status" request?
  F892  Yes, then determine status >>F8FD
  F894  Is command a "read" request?
  F895  Yes, then continue on >>F89A
  F897  Prepair data for write (prenibblize) <FDF0>
  F89A  ---
  F89C  Initialize "retry" count at 64 (FB69)
  F89F  ---
  F8A1  Read an address field - Good read? <FB98>
  F8A4  Yes, then continue on >>F8C0
  F8A6  Decrement "retry" count - More to try? (FB69)
  F8A9  Yes, then try again >>F89F
  F8AB  No, just in case indicate "I/O Error"
  F8AD  Decrement "recalibration" count - More to try? (FB6A)
  F8B0  No, then exit with error >>F8EC
  F8B2  Get "current" track (FB5A)
  F8B5  Preserve it
  F8B6  Double it and
  F8B7  add 16 to it for recalibration
  F8B9  Reinitialize Retry Count
  F8BE  Branch always taken >>F8CE
  F8C3  Was the right track found? (FB5A)
  F8C6  Yes, then continue on >>F8D7
  F8C8  Get "current" track (FB5A)
  F8CB  Preserve it
  F8CC  Get track we found
  F8CD  Double it
  F8CE  Put new value in Device Track Table <FCD3>
  F8D1  Get track we want
  F8D2  And go there <F90C>
  F8D5  Branch always taken >>F89F
  F8DA  Was the right sector found? (FB57)
  F8DD  No, then try again >>F8A6
  F8E1  Is command a "write" request?
  F8E2  Yes, then go do it >>F8F4
  F8E4  Read the data - Good read? <FBFD>
  F8E7  No, then try again >>F8A6
  F8E9  Indicate no errors
  F8EB  BNE Instruction, never taken
  F8EC  Indicate error
```

```
Disk II Device Driver -- V1.0.1 -- 1 JAN 84     NEXT OBJECT ADDR: F8ED
-------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------

  F8ED  Preserve error number (FB58)
  F8F0  Turn motor off (C088)
  F8F3  Return to caller

F8F4 ********** HANDLE WRITE REQUEST **********************************

  F8F4  Write data - Good write? <FD00>
  F8F7  Yes, then exit >>F8E9
  F8F9  Indicate "Write-protect error"
  F8FB  Branch always taken >>F8EC

F8FD ********** GET STATUS ***************************************************

  F8FD  Get Slot number
  F902  Check "write-protect" status (C08E)
  F905  Put result in Carry flag
  F906  Select read mode (C08C)
  F909  Exit with appropriate status >>F8F7

F90C ********** LOCATE DESIRED TRACK **********************************

  F90C  Double the track number for proper phase
  F90D  Preserve destination track * 2 (FB6F)
  F910  Turn all phases off <F925>
  F913  Get offset into Device Track Table <FCF1>
  F916  Get track (FB59)
  F919  Update "current" track (FB5A)
  F91C  Get destination track (FB6F)
  F91F  Update Device Track Table (FB59)
  F922  Move arm to desired track <F933>
  F925  Initialize phase number, starting with 3
  F927  ---
  F928  Clear a phase <F98A>
  F92B  Decrement phase number - More to do?
  F92C  Yes, then continue until all phases done >>F927
  F92E  Divide track number by 2 (FB5A)
  F932  Return to caller

F933 ********** ARM MOVE ROUTINE ************************************

  F933  Preserve track to find (FB72)
  F936  Are we already there? (FB5A)
  F939  Yes, then set appropriate phase and exit >>F987
  F93D  Initialize phase count (halftracks) (FB6B)
  F943  Preserve "current" track for comparisons (FB71)
  F946  Subtract track to find to compute delta-tracks
  F947  Are we already there? (FB72)
  F94A  Yes, then clear prior phase and exit >>F983
  F94C  Positive delta-tracks - go move arm out >>F955
  F94E  Negative delta-tracks - Get absolute value delta-tracks less 1
  F950  Increment current phase to move in (FB5A)
```

```
Disk II Device Driver -- V1.0.1 -- 1 JAN 84     NEXT OBJECT ADDR: F953
-------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------

  F953  Branch always taken >>F95A
  F955  Compute absolute value delta-tracks less 1
  F957  Decrement current phase to move out (FB5A)
  F95A  Compair delta-tracks with phases moved (FB6B)
  F95D  Use smaller value for offset to delay tables >>F962
  F962  Are we pointing at last table value yet?
  F964  Yes, then continue to use current offset  >>F968
  F966  Else, use new offset
  F967  Set Carry flag for set phase operation
  F968  Set a phase <F987>
  F96B  Get delay value from table (FB73)
  F96E  Delay <FB85>
  F971  Get prior phase number (FB71)
  F974  Clear Carry flag for clear phase operation
  F975  Clear a phase <F98A>
  F978  Get delay value from table (FB7C)
  F97B  Delay <FB85>
  F97E  Increment phases moved (FB6B)
  F983  Delay <FB85>
  F987  Get "current" phase number (FB5A)
  F98A  Use low two bits only, zero to three - 000000PP
  F98C  Multiply by two and bring in Carry   - 00000PPC
  F98D  Merge in slot number                 - 0SSS0PPC
  F98F  Put in X-reg for following operation
  F990  Toggle appropriate phase (C080)
  F993  Restore slot number to X-reg
  F995  Return to caller

F996 ********** TABLE 1 ***********************************************
          Read Translate Table with Prenibblize
          Bit mask Tables and Epilog Table in
          unused areas

          Read Translate

          Bit Mask 1
  F9A0  0000000
  F9A1  0000000
  F9A2  1000000
  F9A3  1000000

          Read Translate

          Bit Mask 2
  F9C0  0000000
  F9C1  0100000
  F9C2  0010000
  F9C3  0110000
```

```
Disk II Device Driver -- V1.0.1 -- 1 JAN 84     NEXT OBJECT ADDR: F9C3
-------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------

          Epilog Table ($DE,$AA,$EB)

          Read Translate

          Bit Mask 3
  F9E0  0000000
  F9E1  0001000
  F9E2  0000100
  F9E3  0001100

          Read Translate

FA00 ********** TABLE 2 ***********************************************

          Write Translate Table
                  Every 4th byte starting at $FA03

          Postnibblize Bit mask Tables
                  Bit mask 1 (Every 4th byte starting at $FA00
                  Bit mask 2 (Every 4th byte starting at $FA01
                  Bit mask 3 (Every 4th byte starting at $FA02

  FA00  Entry for Bit Mask 1
  FA01  Entry for Bit Mask 2
  FA02  Entry for Bit Mask 3
  FA03  Entry for Write Translate

FB00 ********** AUXILIARY BUFFER ***************************************

  FB00  Auxiliary Buffer ($56 bytes) >>0056

FB56 ********** VARIABLE AREA ******************************************

  FB56  Track number
  FB57  Sector number
  FB58  Error number

          Disk Device Track Table
FB59      Table Entry
  FB59  Current Unit
  FB5A  Current Track
  FB5B  Slot 1, Devices 1 & 2
  FB5D  Slot 2, Devices 1 & 2
  FB5F  Slot 3, Devices 1 & 2
  FB61  Slot 4, Devices 1 & 2
  FB63  Slot 5, Devices 1 & 2
  FB65  Slot 6, Devices 1 & 2
  FB67  Slot 7, Devices 1 & 2
```

```
Disk II Device Driver -- V1.0.1 -- 1 JAN 84      NEXT OBJECT ADDR: FB67
-------------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------------

  FB69  Retry count (initially 64)
  FB6A  Recalibration count (initially 1)
FB6B    Counter for Read Address routine
FB6B    Temporary storage for Read Address routine
  FB6B  Track counter for Arm Move routine
  FB6C  Checksum computation
  FB6D  Volume found
  FB6E  Sector found
FB6F    Delay counter (low byte)
  FB6F  Track found
FB70    Checksum found
  FB70  Delay counter (high byte)
  FB71  Prior Track
  FB72  Track Number for Arm Move routine

FB73 ********** PHASEON/PHASEOFF TABLES ******************************

         Phase on table (delays for disk head acceleration)

         Phase off table (delays for disk head deacceleration)

FB85 ********** WAIT ROUTINE ***************************************

  FB85  Wait about 100 times A-register (microseconds)
  FB87  ---
  FB92  ---
  FB97  Return to caller

FB98 ********** READ ADDRESS FIELD **********************************

  FB98  Initialize "must find" count at $FCFC
  FB9D  Increment count (low order byte) - Zero yet?
  FB9E  No, skip anead >>FBA5
  FBA0  Increment count (high order byte) - Zero yet? (FB6B)
  FBA3  Yes, exit and indicate Read Error >>FBFB
  FBA5  Read data register (C08C)
  FBA8  Loop until data valid >>FBA5
  FBAA  Is it first address mark ($D5)?
  FBAC  No, then increment "must find" count >>FB9D
  FBAE  Delay for data register to clear
  FBAF  Read data register (C08C)
  FBB2  Loop until data valid >>FBAF
  FBB4  Is it second address mark ($AA)?
  FBB6  No, then see if it's first address mark >>FBAA
  FBB8  Initialize count for four byte read
  FBBA  Read data register (C08C)
  FBBD  Loop until data valid >>FBBA
  FBBF  Is it third address mark ($96)?
  FBC1  No, then see if it's first address mark >>FBAA
  FBC3  Set Interrupt flag
```

```
Disk II Device Driver -- V1.0.1 -- 1 JAN 84      NEXT OBJECT ADDR: FBC4
-------------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------------

  FBC4  Initialize checksum
  FBC9  Read "odd" encoded byte     1X1X1X1X (C08C)
  FBCE  Align "odd" bits            X1X1X1X1
  FBCF  Save for later (FB6B)
  FBD2  Read "even" encoded byte    1X1X1X1X (C08C)
  FBD7  Combine bytes               XXXXXXXX (FB6B)
  FBDA  Preserve data (Volume,Track,Sector,Checksum) (FB6D)
  FBDD  Do checksum computation (FB6C)
  FBE0  Decrement counter - Finished field yet?
  FBE1  No, do some more >>FBC6
  FBE3  Is checksum computation zero?
  FBE4  No, then exit with carry set >>FBFB
  FBE6  Read data register (C08C)
  FBE9  Loop until data valid >>FBE6
  FBEB  Is it first trailing byte ($DE)?
  FBED  No, then exit with carry set >>FBFB
  FBEF  Delay for data register to clear
  FBF0  Read data register (C08C)
  FBF3  Loop until data vaild >>FBF0
  FBF5  Is it second trailing byte ($AA)?
  FBF7  No, then exit with carry set >>FBFB
  FBF9  Clear the Carry flag (no error)
  FBFA  Return to caller
  FBFB  Set the Carry flag (error occured)
  FBFC  Return to caller

FBFD ********** READ DATA (ON THE FLY) ROUTINE  **********************

  FBFD  Convert slot number to an
  FBFE  absolute reference (i.e. $60 -> $EC)
  FC00  Modify code for current slot number (FC5A)
  FC03  (i.e. $C08C,X -> $C0EC) (FC73)
  FC0F  Get data buffer pointers
  FC13  Modify code for current Buffer address (FCAF)
  FC16  Provides access to top 3rd of Buffer (FCB0)
  FC1A  Subtract $54 from current address
  FC1F  Modify code for current address - $54 (FC97)
  FC22  Provides access to middle 3rd of Buffer (FC98)
  FC26  Subtract $57 from current address
  FC2B  Modify code for current address - $AB (FC70)
  FC2E  Provides access to bottom 3rd of Buffer (FC71)

  FC31  Initialize must find count at $20
  FC33  Decrement count - More to do?
  FC34  No, then exit >>FC6D
  FC36  Read data register (C08C)
  FC39  Loop until data valid >>FC36
  FC3B  Is is 1st header mark ($D5)?
  FC3D  No, then try again >>FC33
  FC3F  Delay for register to clear
  FC40  Read data register (C08C)
```

```
Disk II Device Driver -- V1.0.1 -- 1 JAN 84     NEXT OBJECT ADDR: FC43
------------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
------------------------------------------------------------------------

  FC43  Loop until data valid >>FC40
  FC45  Is is 2nd header mark ($AA)?
  FC47  No, then see if it is 1st header mark >>FC3B
  FC49  Delay for register to clear
  FC4A  Read data register (C08C)
  FC4D  Loop until data valid >>FC4A
  FC4F  Is is 3rd header mark ($AD)?
  FC51  No, then see if it is 1st header mark >>FC3B
  FC53  Initialize offset into data buffer
  FC57  Initialize checksum
  FC59  Read a data byte (C0EC)
  FC5E  Translate it (F900)
  FC61  Store it in Auxiliary buffer (FA56)
  FC64  Compute running checksum
  FC66  Increment offset - More to do?
  FC67  Yes, then continue >>FC57
  FC69  Reinitialize offset into data buffer
  FC6B  Branch always taken >>FC72
  FC6D  Set carry flag indicating error
  FC6E  Return to caller
  FC6F  Store byte in Primary buffer (bottom third) (1000)
  FC72  Read a data byte (C0EC)
  FC77  Translate it and merge in (F900)
  FC7A  bits from Auxiliary buffer (FA56)
  FC80  Increment offset - done yet?
  FC81  No, then do another >>FC6F
  FC83  Save last byte for later, no time now
  FC84  Strip off last two bits   XXXXXX00
  FC86  Reinitialize offset
  FC88  Read a byte (C0EC)
  FC8D  Translate it and merge in  (F900)
  FC90  bits from Auxiliary buffer (FA56)
  FC96  Store byte in Primary buffer (middle third) (1000)
  FC99  Increment offset - done yet?
  FC9A  No, then do another >>FC88
  FC9C  Read a byte (C0EC)
  FCA1  Strip off last two bits   XXXXXX00
  FCA3  Reinitialize offset
  FCA5  Translate byte and merge in (F900)
  FCA8  bits from Auxiliary buffer (FA54)
  FCAE  Store byte in Primary buffer (top third) (1000)
  FCB1  Read a byte (C0EC)
  FCB6  Increment offset - done yet?
  FCB7  No, then do another >>FCA5
  FCB9  Strip off last two bits   XXXXXX00
  FCBB  Is checksum valid? (F900)
  FCBE  No, then exit with error >>FCCC
  FCC0  Get slot number
  FCC2  Read data register (C08C)
  FCC5  Loop until data valid >>FCC2
  FCC7  Is is 1st trailing mark ($DE)?
```

```
Disk II Device Driver -- V1.0.1 -- 1 JAN 84     NEXT OBJECT ADDR: FCCA
------------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
------------------------------------------------------------------------

  FCCA  Yes, then continue with carry clear >>FCCD
  FCCC  Set Carry flag indicating error
  FCCD  Get byte we stored away, we have time now
  FCCE  Set proper offset
  FCD0  Store byte in Primary buffer (offset $55)
  FCD2  Return to caller

FCD3 ********** UPDATE DEVICE TRACK TABLE ****************************

  FCD3  Get offset into Device Track Table <FCF1>
  FCD6  Update Device Track Table (FB59)
  FCD9  Return to caller

FCDA ********** DETERMINE IF DRIVE IS ON (DATA CHANGING)**************

  FCDA  Get slot number
  FCDC  Initialize counter
  FCDE  Read data register (C08C)
  FCE1  Delay 25 cycles <FCF0>
  FCE6  Has data register changed? (C08C)
  FCE9  Yes, then exit >>FCF0
  FCEB  Just in case indicate No Device Connected Error
  FCED  Decrement count - 256 tries yet?
  FCEE  No, try again >>FCDE
  FCF0  Return to caller

FCF1 ********** CONVERT SLOT/DRIVE TO TABLE OFFSET *          SKP  1***
  FCF1  Preserve A-register
  FCF2  Get Unit number        DSSS0000
  FCF4  Divide by 16           0000DSSS
  FCF8  Put Drive into Carry   0000DSSS  D
  FCFA  Strip out Drive        00000SSS  D
  FCFC  Roll left              0000SSSD
  FCFD  Put result in X-register
  FCFE  Restore A-register
  FCFF  Return to caller

FD00 ********** WRITE DATA ROUTINE ************************************

  FD00  Set Carry flag (anticipate error)
  FD04  Is diskette "write-protected"? (C08E)
  FD07  No, then continue on >>FD0C
  FD09  Go to error routine >>FDDF
  FD0C  Put transition byte from secondary buffer (FB00)
  FD0F  into zero page for timing
  FD11  Use $FF for "sync" byte
  FD13  Write first "sync" byte (C08F)
  FD19  Set counter for four more
  FD1C  Delay so that writes occur
  FD1D  Exactly on 40 cycle loops
  FD1E  ---
```

```
Disk II Device Driver -- V1.0.1 -- 1 JAN 84     NEXT OBJECT ADDR: FD20
-------------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------- ----------------------------------------

  FD20  Write "sync" byte <FDE7>
  FD23  Decrement counter, done yet?
  FD24  No, then do another >>FD1E
  FD26  Write first data mark ($D5)
  FD2B  Write second data mark ($AA)
  FD30  Write third data mark ($AD)
  FD35  Initialize checksum
  FD36  Initialize index into Auxiliary buffer
  FD38  Branch always taken >>FD3D
  FD3A  Get data byte (Auxiliary buffer) (FB00)
  FD3D  Exclusive-or with previous data byte (FAFF)
  FD40  Put result in X-reg for table lookup
  FD41  Lookup "disk byte" in table (FA03)
  FD44  Get slot
  FD46  Write "disk byte" (C08D)
  FD4C  Decrement index - Done with Auxiliary buffer?
  FD4D  No, then another byte >>FD3A
  FD4F  Get last byte of Auxiliary buffer
  FD51  Initialize index into Primary buffer
  FD53  Exclusive-or with next data byte (1000)
  FD56  Strip out last two bits  XXXXXX00
  FD58  Put result in X-reg for table lookup
  FD59  Lookup "disk byte" in table (FA03)
  FD5C  Get slot
  FD5E  Write "disk byte" (C08D)
  FD64  Get data byte (Primary buffer) (1000)
  FD67  Increment offset, end of this page?
  FD68  No, then continue on >>FD53
  FD6A  Did buffer start on page boundary?
  FD6C  Yes, then go write checksum  >>FDC0
  FD6E  Did buffer start one past page boundary?
  FD70  Yes, then go write last byte >>FDB3
  FD72  Carry indicates odd or even buffer end
  FD73  Get transition byte
  FD75  Write it (C08D)
  FD7B  Get second transition byte
  FD7D  Delay 2 cycles for correct timing
  FD7E  Increment offset, buffer end on odd byte?
  FD7F  Yes, go see if we're done then >>FD99
  FD81  Exclusive-or with next data byte (1100)
  FD84  Strip out last two bits  XXXXXX00
  FD86  Put result in X-reg for table lookup
  FD87  Lookup "disk byte" in table (FA03)
  FD8A  Get slot
  FD8C  Write "disk byte" (C08D)
  FD92  Get data byte (Primary buffer - page 2) (1100)
  FD95  Increment offset
  FD96  Exclusive-or with next data byte (1100)
  FD99  End of buffer? - Put result in carry
  FD9B  Strip out last two bits  XXXXXX00
  FD9D  Put result in X-reg for table lookup
```

```
Disk II Device Driver -- V1.0.1 -- 1 JAN 84     NEXT OBJECT ADDR: FD9E
-------------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------------

  FD9E  Lookup "disk byte" in table (FA03)
  FDA1  Get slot
  FDA3  Write "disk byte" (C08D)
  FDA9  Get data byte (Primary buffer - page 2) (1100)
  FDAC  Increment offset - Done yet?
  FDAD  No, then do another >>FD81
  FDAF  Yes, then go write checksum >>FDB1
  FDB1  --- >>FDC0
  FDB3  Get last byte (003B)
  FDB6  Write it (C08D)
  FDBC  Delay 14 cycles for correct timing
  FDC0  Use last byte in Primary buffer as checksum
  FDC2  Lookup "disk byte" (FA03)
  FDC5  Get slot
  FDC7  Write "disk byte" (C08D)
  FDCD  Initialize offset into "epilog" table
  FDCF  Delay 11 cycles for correct timing
  FDD3  Load "epilog" from table ($DE,$AA,$EB,$FF) (F9C4)
  FDD6  Go write it <FDE9>
  FDD9  Increment offset
  FDDA  Done all four yet?
  FDDC  No, then do another >>FDD3
  FDDE  Clear Carry flag (no error)
  FDDF  Select read mode (C08E)
  FDE5  Return to caller

FDE6 ********** WRITE A BYTE SUBROUTINE ******************************

  FDE6  Wait 9 cycles before write
  FDE7  Wait 7 cycles before write
  FDE9  Put A-register in data register (C08D)
  FDEC  And write data register (C08C)
  FDEF  Return to caller

FDF0 ********** PRENIBLIZE BLOCK ROUTINE ******************************

  FDF0  Get buffer pointer
  FDF5  Add $2 to buffer address
  FDF7  To access top third of buffer >>FDFA
  FDFA  Store result in code below (FE30)
  FE01  Subtract $54 from buffer address
  FE03  To access middle third of buffer >>FE06
  FE06  Store result in code below (FE25)
  FE0D  Subtract $AA from buffer address
  FE0F  To access bottom third of buffer >>FE12
  FE12  Store result in code below (FE1B)
  FE18  Initialize offset
  FE1A  Get data byte (bottom third)    XXXXXXXX (1000)
  FE1D  Get last two bits               000000AB
  FE1F  Put in X-reg for table lookup
  FE20  Use lookup to reposition bits   0000BA00 (F9E0)
```

```
Disk II Device Driver -- V1.0.1 -- 1 JAN 84     NEXT OBJECT ADDR: FE23
------------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
------------------------------------------------------------------------

  FE23  Save result on stack
  FE24  Get data byte (middle third)     XXXXXXXX (1056)
  FE27  Get last two bits                000000CD
  FE29  Put in X-reg for table lookup
  FE2A  Get current value from stack     0000BA00
  FE2B  Merge in new bits using table    00DCBA00 (F9C0)
  FE2E  Save result on stack
  FE2F  Get data byte (top third)        XXXXXXXX (10AC)
  FE32  Get last two bits                000000EF
  FE34  Put in X-reg for table lookup
  FE35  Get current value from stack     00DCBA00
  FE36  Merge in new bits using table    FEDCBA00 (F9A0)
  FE39  Save result on stack
  FE3A  Get offset into primary buffer
  FE3B  Compute offset into Auxiliary buffer
  FE3D  Put in X-reg
  FE3E  Get data byte just created       FEDCBA00
  FE3F  Store it in Auxiliary buffer (FB00)
  FE42  Increment offset primary buffer, done yet?
  FE43  No, then do another >>FE1A
  FE45  Get low order byte of buffer
  FE47  Subtract 1 (offset to last byte in buffer)
  FE48  Save it for later
  FE4A  Get low order byte of buffer
  FE4C  Modify code in Write Data Routine (offset) (FD52)
  FE4F  Buffer on page boundary? - Yes, skip ahead >>FE5F
  FE51  Else, compute offset to last byte
  FE53  Before page boundary
  FE54  Get byte (page boundary -1)
  FE56  Point at next byte (page boundary)
  FE57  Exclusive-or them together       XXXXXXXX
  FE59  Strip off last two bits          XXXXXX00
  FE5B  Put in X-reg for table lookup
  FE5C  Get "disk byte" from table (transition byte) (FA03)
  FE5F  Save result (0 indicates page boundary)
  FE61  Buffer on page boundary? - Yes skip ahead >>FE6F
  FE63  Get offset to last byte in buffer
  FE65  Carry indicates odd or even buffer start
  FE66  Get byte (page boundary)
  FE68  Did buffer start on odd byte? - Yes skip >>FE6D
  FE6A  Point at next byte (page boundary +1)
  FE6B  Exclusive-or them together
  FE6D  Save result
  FE6F  Point at last byte in buffer
  FE71  Get last byte in buffer          XXXXXXXX
  FE73  Strip off last two bits          XXXXXX00
  FE75  Save result ("checksum byte")
  FE77  Get high order byte of buffer
  FE79  Modify code in Write Data Routine (FD55)
  FE8C  Get slot number for this operation
  FE8E  Modify code in Write Data Routine (FD5D)
```

```
Disk II Device Driver -- V1.0.1 -- 1 JAN 84     NEXT OBJECT ADDR: FE9A
------------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
------------------------------------------------------------------------

  FE9A   Return to caller

FE9B ********** DETERMINE IF SLOT/DRIVE HAS CHANGED ******************

  FE9B  Compair unit number with "current" unit number (FB59)
  FE9E  Put "current" drive in Carry
  FE9F  Has slot changed? - No, then exit >>FEBD
  FEA9  Get "current" slot
  FEAB  Put in X-register
  FEAC  Exit if Slot 0 >>FEBD
  FEAE  Is "current" motor is on? <FCDC>
  FEB1  No, then exit >>FEBD
  FEB8  Wait until "current" motor is off (FB70)
  FEBB  Or else timeout >>FEA6
  FEBD  Return to caller
  FEBE  Unused >>0042
003B
```

```
IRQ Handler -- V1.0.1 -- 1 JAN 84                NEXT OBJECT ADDR: FF9B
------------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
------------------------------------------------------------------------

FF9B    MODULE STARTING ADDRESS
            **********************************
            *                                *
            * Interrupt handler:             *
            *     Resides at $FF9B           *
            *                                *
            *                                *
            *                                *
            *                                *
            * Versions 1.0.1 -- 1 JAN 84     *
            **********************************

FF9B ********** GLOBAL PAGE EQUATES **********************************

BF56    Temporary storage 1
BF57    Temporary storage 2
BF88    A register savearea
BF8D    Bank ID byte
BFD3    IRQ exit code

FF9B ********** EXTERNAL EQUATES ************************************

D000    RAM/ROM test byte
C082    ROM Select
C08B    Bank1 Select

FF9B ********** IRQ CODE ********************************************

  FF9B  Put A-Register on stack
  FF9C  Get Accumulator value from $45
  FF9E  and save it (BF56)
  FFA1  Replace $45 with A-Register
  FFA2  since it may have been destroyed
  FFA4  Load Status register
  FFA5  Restore onto stack
  FFA6  Isolate B flag - Was it a BRK?
  FFA8  Yes, skip Interrupt stuff >>FFC2
  FFAA  Else, Check location $D000 (D000)
  FFAD  Do we have RAM active
  FFAF  Yes, indicate so >>FFB3
  FFB1  Else, indicate ROM
  FFB3  Update Bank ID byte (BF8D)
  FFB6  Also save temporarily (BF57)
  FFB9  Push ($BF50) address of
  FFBB  routine to bank in Ram and
  FFBC  call IRQ on the stack
  FFBF  Push a new P-Register on stack with
  FFC1  the Interrupt Disable flag set
  FFC2  Push ($FA41) address less 1 of
  FFC4  Monitor IRQ on the stack
  FFC8  Select ROM - execution continues in ROM (C082)
```

```
IRQ Handler -- V1.0.1 -- 1 JAN 84                NEXT OBJECT ADDR: FFC8
------------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
------------------------------------------------------------------------

            ******* RESET CODE ***************

  FFCB  Push ($FA61) address less 1 of (FFD7)
  FFCE  Hardware Reset routine on to stack
  FFD3  Exit via select ROM code above >>FFC8

  FFD6  Address (-1) of Hardware Reset routine

            ******* IRQ CODE *****************
          Called via $BF50 in System Global Page

  FFD8  Save Accumulator in Global page (BF88)
  FFDB  Restore $45 with original value (BF56)
  FFE0  Select RAM (read & write) (C08B)
  FFE3  use Bank 1 (C08B)
  FFE6  Get Bank ID byte (BF57)
  FFE9  Leave via Global Page IRQ exit code >>BFD3

FFEC ********** UNUSED ************************************************

  FFEC  --- >>000E

FFFA ********** VECTORS ***********************************************

  FFFA  NMI Vector
  FFFC  Reset Vector
  FFFE  IRQ Vector
```

```
BI Relocator -- V1.0.1 -- 1 JAN 84              NEXT OBJECT ADDR: 2000
-------------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------------

2000    MODULE STARTING ADDRESS

          ****************************************
          *                                      *
          *  PRODOS BASIC INTERPRETOR RELOCATOR  *
          *       LOADED AS THE FIRST TWO BLOCKS *
          *       OF BASIC.SYSTEM AT $2000.      *
          *       THIS ROUTINE MOVES THE BASIC   *
          *       INTERPRETOR TO HIGH MEMORY.    *
          *                                      *
          *  VERSION 1.0.1 -- 1 JAN 84           *
          *                                      *
          ****************************************

          ******** ZERO PAGE ADDRESSES *************

0000    FROM POINTER FOR COPY
0001
0002    TO POINTER FOR COPY
0003
0036    CSWL VECTOR
0038    KSWL VECTOR
006F    APPLESOFT START OF STRINGS
0073    APPLESOFT HIMEM
00F2    APPLESOFT TRACE FLAG

          ******** EXTERNAL ADDRESSES **************

0200    PATHNAME BUFFER
0280    PREFIX BUFFER
0281    START OF PREFIX NAME

03D0    WARMSTART VECTOR
03D3    COLDSTART VECTOR
03F0    BRK HANDLER ADDRESS
03F1
03F2    RESET HANDLER ADDRESS
03F3
03F4    POWER-UP BYTE
03F5    APPLESOFT & VECTOR
03F8    CTL-Y VECTOR

          ******** SCREEN LINE ADDRESSES **********

0400    FIRST SCREEN BUFFER LINE
0480    SCREEN BUFFER LINE
0628    SCREEN BUFFER LINE
```

```
BI Relocator -- V1.0.1 -- 1 JAN 84              NEXT OBJECT ADDR: 2000
-------------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------------

          ******** BASIC GLOBAL PAGE **************

BE00    BASIC INTERPRETOR ENTRY POINT
BE03    BI COMMAND SCANNER (SYNTAX)
BE10    COUT VECTORS FOR EACH SLOT
BE20    KSWL VECTORS FOR EACH SLOT
BE3C    DEFAULT SLOT NO.
BE3D    DEFAULT DRIVE NO.
BEFB    HIMEM

          ******** SYSTEM GLOBAL PAGE *************

BF00    MACHINE LANGUAGE INTERFACE ENTRY
BF30    LAST DEVICE USED
BF58    MEMORY MAP
BF98    MACHINE TYPE FLAGS
BF99    SLOTS WHICH CONTAINS CARDS WITH ROM
BF9A    IF 0, NO PREFIX ACTIVE

          ******** I/O PORT ADDRESSES *************

C00C    TURN OFF 80 COLUMN BOARD

          ******** ROM ADDRESSES ******************

E000    APPLESOFT ENTRY POINT
FA59    BRK HANDLER
FB2F    INIT SCREEN, MONITOR, ETC.
FC58    CLEAR SCREEN, HOME CURSOR
FDF0    CHARACTER OUTPUT TO SCREEN
FE84    SET NORMAL CHARACTER ATTRIBUTE

2000 *********** BASIC INTERP RELOCATOR ENTRY ************************

  2000  $00 --> $2400
  2004  $02 --> $9A00
  200E  COPY 35 PAGES
  2011  COPY INTERP TO HIGH MEMORY AT $9A00 <207B>
  2016  PAGE FOLLOWING INTERP IMAGE IS...
  2018  BASIC GLOBAL PAGE IMAGE
  201A  COPY THAT TO $BE00 <207B>
  201D  TURN 80 COLUMNS OFF (C00C)
  2020  SET NORMAL CHARACTER ATTRIBUTE <FE84>
  2023  INITIALIZE SCREEN/WINDOW <FB2F>
  2026  CLEAR SCREEN/HOME CURSOR <FC58>
  202D  SET BITMAP TO MARK LOWER 48K FREE (BF58)
  2033  EXCEPT PAGES 0 AND 1 AND
  2035   TEXT PAGES 4 THROUGH 7 (BF58)
  203D  MARK $9000-$BFFF IN USE..
  2048  EXCEPT FOR $BA00-$BDFF ARE FREE
```

```
BI Relocator -- V1.0.1 -- 1 JAN 84              NEXT OBJECT ADDR: 204D
-------------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------------

  204D  LOOK AT LANGUAGE IN ROM  (E000)
  2050  IS IT APPLESOFT?
  2052  NO, THEN CAN'T RUN INTERP >>2068
  2057  GOT AT LEAST 64K?
  2059  ND, THIS ONLY WDRKS IN 64K >>2068
  205D  SET MY CSWL/KSWL FOR INTERP INIT (21CB)
  2063  COPY ALL 4 BYTES >>205D
  2065  THEN GO TO BASIC COLDSTART >>E000
          (WE WILL GET CONTROL AT 208B AGAIN)

2068 *********** ERROR EXIT ****************************************

  2068  ---
  206A  PRINT "UNABLE TO EXECUTE BASIC SYSTEM" (21F8)
  2073  ALLOW REBDOT IF RESET PRESSED (03F4)
  2079  GO TO SLEEP FOREVER >>2079

207B *********** CDPY PAGES ($0/1-->$2/3) ****************************

  207B  ---
  207C  COPY FROM $0/1
  207E  TO $2/3
  2081  A PAGE AT A TIME >>207B
  2087   COUNT PAGES
  208A  RETURN

208B *********** CSWL INTERCEPT / CONTINUE ***************************

  208B  "]" APPLESOFT PROMPT?
  208D  NO...DON'T PRINT WHATEVER IT IS >>208A
  208F  YES, APPLESOFT DONE SETTING UP (BE10)
  2092  POINT CSWL TO STANDARD DUTPUT
  2099  CHECK LAST DEVICE USED (BF30)
  209C  SET ONLINE PARAMETER TO THIS (21F1)
  20A2  DRIVE ONE OR TWO? >>20A5
  20A5  STORE DEFAULT DRIVE (D) (BE3D)
  20A9  ISOLATE SLOT FROM DEVICE ND.
  20AE  AND STORE DEFAULT SLDT (S) (BE3C)
  20B5  GET SLOT BYTE SHOWING CARDS PRESENT (BF99)
  20B9  PICK DFF ITS BITS ONE BY ONE
  20BF  SET OUTVECS AND INVECS TO $CS00 (BE10)
  20C2  FOR ALL SLOTS WITH ROMS IN THEM (BE20)
  20CC  ---
  20D2  SET HIMEM TO $9600
  20D4  IN VARIOUS PLACES
  20DB  GOT A DEFAULT PREFIX? (BF9A)
  20DE  NO >>2105
  20E0  YES, MLI: GET PREFIX <BF00>
  20E6  ERROR? >>2142
  20ED  BACKSCAN PREFIX FOR "/"'S (0280)
  20F2  AND COUNT THEM IN $21EE (21F7)
```

```
BI Relocator -- V1.0.1 -- 1 JAN 84              NEXT OBJECT ADDR: 20F5
-------------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------------

  20F5  ---
  20F6  FOR A CDUNT OF SUBLEVELS >>20ED
  20FD  MORE THAN JUST VOLUME NAME? >>2126
  20FF  NO, MLI: SET PREFIX <BF00>
  2105  MLI: ONLINE <BF00>
  210B  ERROR? >>2142
  210D  GET VOL NAME LENGTH (0281)
  2112  NONE THERE? >>2142
  2116  ADD ONE TO NAME LENGTH (0280)
  211B  AND PREFIX IT WITH A "/" (0281)
  211E  MLI: SET PREFIX <BF00>
  2124  ERROR? >>2142

        ******** FIND STARTUP FILE **************

  2126  MLI: GET_FILE_INFO <BF00>
  2129  FIND "STARTUP" FILE
  212C  ERROR? >>2142
  2131  SAVE LENGTH OF STARTUP FILE NAME (21EF)
  2134  COPY NAME TO $200 (21E5)
  213D  FIRST CDMMAND WILL BE "-STARTUP"
  2142  CHECK NUMBER OF SUBLEVELS (21F7)
  2147  MORE THAN JUST VDL? >>214F
  2149  MLI: SET PREFIX <BF00>
  214F  ANY STARTUP FILE NAME? (21EF)
  2152  YES, SKIP MESSAGE >>2178
  2154  SET TRUE KSWL <21BA>
  2159  PRINT '          PRODDS BASIC 1.0' (2220)
  2164  PRINT '       COPYRIGHT ... (223C)
  216D  SKIP THREE LINES

        ******** FINISH UP AND GO TD BI *********

  2178  ---
  217A  COPY WARMSTART JMP TO 3PAGE (21B0)
  2180  AND COLDSTART (03D3)
  2183  AND CTL-Y (03F8)
  2186  PDINT & VECTOR (21B7)
  2189  TO $BE03 (CMD SCANNER) (03F5)
  218F  CDPY BRK HANDLER JMP ALSO (21B3)
  219E  AND RESET JMP (03F2)
  21A9  SET POWER-UP BYTE ACCORDINGLY (03F4)
  21AE  SET APPLESOFT IN NON-TRACE MODE
  21B0  AND GO TO INTERPRETER >>BE00

        ******** VECTOR ADDRESSES ***************
```

```
BI Relocator -- V1.0.1 -- 1 JAN 84              NEXT OBJECT ADDR: 21B3
-------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------

  21B3  BREAK HANDLER ADDRESS FOR 3PAGE
  21B5  RESET HANDLER IS BASIC INTERP
  21B7  APPLESOFT & GOES TO BI CMD SCANNER >>BE03

21BA ********** FIRST KSWL INTERCEPT *********************************

  21BA  SET KSWL TO CURRENT DEVICE HANDLER (BE20)
  21C4  RETURN LENGTH OF FIRST COMMAND (21E5)
  21C8  FOLLOWED BY A RETURN
  21CA  RETURN

21CB ********** DATA ************************************************

  21CB  CSWL (208B) INTERCEPT ADDR
  21CD  KSWL (21B1) INTERCEPT ADDR

  21CF  GET FILE INFO PARMLIST
  21D0  FILE NAME IS AT $21DC

  21E2  SET PREFIX PARM LIST
  21E3  FOR PREFIX AT $21E4

  21E5  STARTUP FILE NAME LENGTH (07)
  21E6  'STARTUP'

  21ED  NULL PREFIX
  21EE  "/"

  21EF  SAVED LENGTH OF STARTUP FILE NAME

  21F0  ONLINE PARM LIST
  21F2  PUT VOLUME NAME AT $281

  21F4  SET PREFIX PARMLIST
  21F5  PREFIX IS AT $280

  21F7  NUMBER OF SUBLEVELS IN PREFIX +1

  21F8  '***           UNABLE TO EXECUTE BASIC SYSTEM  ***'
  2220  '                    PRODOS BASIC 1.0'
  223C  '                COPYRIGHT APPLE, 1983'

225A ********** NOT USED ********************************************

  225A  ---
```

```
BI Relocator -- V1.0.1 -- 1 JAN 84              NEXT OBJECT ADDR: 225C
-------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------

2400 ********** START OF BI IMAGE ***********************************

2400    BASIC INTERP IMAGE
```

```
BASIC Interpreter (BI) -- V1.0.1 -- 1 JAN 84    NEXT OBJECT ADDR: 9A00
------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
------------------------------------------------------------------

9A00    MODULE STARTING ADDRESS

          ****************************************
          *                                      *
          *  PRODOS BASIC INTERPRETER (BI)       *
          *       THIS CODE STARTS IN THE THIRD  *
          *       BLOCK OF THE FILE BASIC.SYSTEM.*
          *       IT PERFORMS COMMAND HANDLING   *
          *       FOR ALL BUILT-IN PRDDOS CDM-   *
          *       MANDS AND SUPPORTS BASIC'S FILE*
          *       HANOLING.                      *
          *                                      *
          *  VERSION 1.0.1 -- 1 JAN 84           *
          *                                      *
          ****************************************

          ******** ZERO PAGE ADORESSES ************

0024    CURSOR HDRIZONTAL
0028    SCREEN LINE BASE ADDR
0029
0033    MONITOR PROMPT CHARACTER
0036    CRT OISPLAY VECTOR (CSWL)
0037
0038    KEYBOARD INPUT VECTDR (KSWL)
0039
003A    SCRATCH PDINTER AND LOOP COUNTER
003B
003C    SCRATCH POINTER AND LDDP CDUNTER
003D
003E    PDINTER TO APPLESOFT VARIABLES
003F
0050    APPLESOFT: LINE NUMBER
0051
0067    APPLESOFT: START OF PROGRAM PTR
0068
0069    APPLESOFT: LOMEM (START OF VARS)
006A
006B    APPLESOFT: START OF ARRAY VARS PTR
006C
006E    APPLESOFT: START OF FREEAREA PTR
006D
006F    APPLESOFT: START OF STRINGS PTR
0070
0073    APPLESOFT: HIMEM (END OF STRINGS)
0074
0075    APPLESOFT: CURRENT LINE BEING EXECUTED
0076
009B    APPLESOFT: ADDR OF LINE AFTER FINOLINE
009C
00AF    APPLESOFT: ENO OF PROGRAM PTR
```

```
BASIC Interpreter (BI) -- V1.0.1 -- 1 JAN 84    NEXT OBJECT ADDR: 9A00
------------------------------------------------------------------
ADDR    DESCRIPTIDN/CONTENTS
------------------------------------------------------------------

00B0
00B8    APPLESOFT: START OF PROGRAM PTR
00B9
00D6    APPLESDFT: PROGRAM LDCKED (PRDTECTEO)
00D8    APPLESOFT: ONERR ACTIVE FLAG
00OE    APPLESOFT: ONERR COOE
00F2    APPLESDFT: TRACE ACTIVE FLAG
00F8    APPLESOFT: INTERNAL STACK

          ******** EXTERNAL ADORESSES *************

0100    START OF 6502 STACK
0200    KEYBOARD INPUT LINE BUFFER
03F4    POWERON RESET FLAG

          ******** BI GLOBAL PAGE *****************

BE06    EXTERNAL COMMAND ENTRY TO BI
BE0C    PRINT ERROR MESSAGE ENTRY TO BI
BE0F    PROOOS ERRDR CODE
BE10    OUTPUT VECTDRS FOR ALL SLOTS
BE30    CURRENT DUTPUT VECTOR
BE32    CURRENT INPUT VECTOR
BE34    PROOOS INTERCEPT VECTORS (INPUT/OUTPUT)
BE38    BI'S INTERNAL REOIRECTION VECTORS
BE3C    OEFAULT SLOT
BE3D    DEFAULT DRIVE
BE3E    A REGISTER SAVE AREA
BE3F    X REGISTER SAVE AREA
BE40    Y REGISTER SAVE AREA
BE41    TRACE FLAG (APPLESOFT TRACE ON/DFF)
BE42    IMMEDIATE COMMANOS=0, OEFERREO=1
BE43    EXEC FILE ACTIVE=$80
BE44    READ FILE ACTIVE=$80
BE45    WRITE FILE ACTIVE=$80
BE46    READING PREFIX ACTIVE=$80
BE47    DIRECTORY FILE BEING ACCESSED
BE49    FREE STRING SPACE OURING GARBAGE COLLECT
BE4A    BUFFERED I/D BYTE COUNT
BE4B    INOEX INTD INPUT COMMAND LINE
BE4C    LAST OUTPUT CHAR TO PREVENT RECURSION
BE4O    NUMBER OF OPEN NON-EXEC FILES
BE4E    EXEC FILE BEING CLOSED FLAG
BE4F    REAO FILE IS TRANSLATEO OIRECTORY
BE50    VECTOR TO EXTERNAL COMMAND HANOLER
BE52    LENGTH-1 OF EXTERNAL COMMANO STRING
BE53    COMMANO NUMBER
BE54    PARAMETERS ALLOWED FOR THIS COMMAND
          (SEE BIT OEFINITIONS IN TABLE LATER)
BE56    PARAMETERS FOUNO WITH THIS COMMANO
          (SAME BIT DEFINITIONS AS FOR PBITS)
```

```
BASIC Interpreter (BI) -- V1.0.1 -- 1 JAN 84    NEXT OBJECT ADDR: 9A00
-------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------

BE58    A KEYWORD VALUE
BE5A    B KEYWORD VALUE
BE5D    E KEYWORD VALUE
BE5F    L KEYWORD VALUE
BE61    S KEYWORD VALUE
BE62    D KEYWORD VALUE
BE63    F KEYWORD VALUE
BE65    R KEYWORD VALUE
BE68    @ KEYWORD VALUE
BE6A    T KEYWORD VALUE
BE6B    SLOT NUMBER FROM IN# OR PR#
BE70    ISSUE MLI CALL AND XLATE ERROR CODES
          MLI PARM LIST FIELDS
BEA3    CREATE: ACCESS CODE
BEA4    CREATE: FILE ID
BEA5    CREATE: AUX ID
BEA7    CREATE: FILE KIND
BEB4    SET/GET FILE INFO: PARM COUNT
BEB7    SET/GET FILE INFO: ACCESS CODE
BEB8    SET/GET FILE INFO: FILE ID
BEB9    SET/GET FILE INFO: AUX ID
BEBB    SET/GET FILE INFO: FILE KIND
BEBC    SET/GET FILE INFO: BLOCKS USED
BEBE    SET/GET FILE INFO: MODIFY DATE/TIME
BEC7    ONLINE/GET/SET MARK/EOF/BUF: REF NUM
BEC8    ONLINE/GET/SET MARK/EOF/BUF: MARK/BUF
BECE    OPEN: SYSTEM BUFFER
BED0    OPEN: REF NUM RETURNED
BED2    NEWLINE: REF NUM
BED3    NEWLINE: NEW LINE CHAR (ALWAYS CR)
BED6    READ/WRITE: REF NUM
BED7    READ/WRITE: DATA ADDRESS
BED9    READ/WRITE: LENGTH OF DATA
BEDB    READ/WRITE: ACTUAL LENGTH TRANSMITTED
BEDE    CLOSE/FLUSH: REF NUM
BEFB    BASIC HIMEM VALUE

          ******** SYSTEM GLOBAL PAGE *************

BF03    QUIT VECTOR
BF30    LAST DEVICE USED
BF58    MEMORY UTILIZATION BIT MAP
BF94    OPEN FILE LEVEL
BF9A    PREFIX ACTIVE FLAG (IF NONZERO)

          ******** INPUT/OUTPUT LOCATIONS *********
```

```
BASIC Interpreter (BI) -- V1.0.1 -- 1 JAN 84    NEXT OBJECT ADDR: 9A00
-------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------

C000    KEYBOARD STROBE
C010    KEYBOARD STROBE CLEAR
CFFF    RESET I/O ROMS

          ******** APPLESOFT ROM LOCATIONS ********

D43F    APPLESOFT RESTART ENTRY
D61A    FIND LINE BY NUMBER IN APPLESOFT
D665    SET POINTERS IN APPLESOFT
D7D2    EXECUTE NEW APPLESOFT STATEMENT
D820    APPLESOFT CMD EXECUTE
D865    APPLESOFT SIGNAL ERROR
ED24    APPLESOFT PRINT DECIMAL NUMBER
F273    APPLESOFT SET NORMAL CHARS

          ******** MONITOR ROM LOCATIONS **********

FC58    MONITOR CLEAR SCREEN/HOME CURSOR
FC9C    MONITOR CLEAR TO EOL
FD10    MONITOR READ KEY (NO CURSOR)
FDED    COUT VECTOR

9A00 *********** BASIC INTERPRETOR LOAD POINT ************************
          (ENTRY POINT IS AT $AC35, WARMDOS)

9A00 *********** REMOVE KSWL/CSWL INTERCEPTS *************************

  9A00  ---
  9A01  REPLACE CSWL/KSWL WITH CURRENT (BE30)
  9A04  ACTUAL DEVICE DRIVER VECTORS
  9A16  RETURN

9A17 *********** RESET MODE/SET BI INTERCEPTS *************************

  9A17  SET IMMEDIATE COMMAND MODE
  9A19  AND GO SET I/O VECTORS <9FAD>
  9A1C  KSWL/H ALREADY SET?
  9A21  NO? THEN CHECK CSWL >>9A26
  9A23  YES, CONTINUE >>9AA3
  9A26  CSWL/H ALREADY SET?
  9A2B  YES, CONTINUE >>9AA3
  9A2D  NO, SAVE CURRENT INTERCEPTS FIRST >>9A8D

9A2F *********** OUTPUT INTERCEPT: MODE = 0 ***************************
          (IMMEDIATE MODE)
```

```
BASIC Interpreter (BI) -- V1.0.1 -- 1 JAN 84    NEXT OBJECT ADDR: 9A2F
-------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------

  9A2F  "#" CHARACTER? (9F98)
  9A32  NO... >>9A54
  9A34  ELSE, SAVE X REG (BE3F)
  9A38. CHECK STACK FOR $D812 AS RETURN ADDR (0103)
  9A3B  (APPLESOFT TRACE, PRINTING #LINENO)
  9A44  NOT TRACING? >>9A6E
  9A46  ELSE, SET DEFERED MODE=4
  9A4B  GET SET TO PRINT THE "#" (9F98)
  9A4E  RESTORE X REG (BE3F)
  9A51  AND GO TO OTHER OUTPUT HANDLER >>B84B

  9A54  NOT A #, SAME AS LAST OUTPUT THO? (BE4C)
  9A57  (SAVE FOR NEXT TIME THRU) (BE4C)
  9A5A  NO, ALL IS WELL >>9A74
  9A5C  TWO RETURNS IN A ROW?
  9A5E  NO, ALL IS WELL >>9A74
  9A60  HAS HORIZONTAL CURSOR POSN CHANGED?
  9A62  YES... >>9A69
  9A64  ELSE, ANYTHING IN PATHNAME BUFFER? (BCBD)
  9A67  (MUST BE ALPHA)
  9A69  RESTORE A REG
  9A6B  PATHNAME IS THERE... >>9A74
  9A6D   ELSE, WE ARE RECURSING INFINITELY, EXIT!
  9A6E  WE WERE'NT TRACING AFTER ALL, RESTORE X (BE3F)
  9A71  AND A REGS, THEN FALL THRU TO EXIT (9F98)

9A74 ********** ECHO OUTPUT CHAR AND EXIT **************************

  9A74  PUT BACK REAL CSWL/KSWL VECTORS <9A00>
  9A77  OUTPUT THE CHARACTER <FDED>
  9A7A  WAS IT A RETURN?
  9A7C  NO, EXIT NOW >>9A8D
  9A7E  ELSE, WAS APPLESOFT TRACING?
  9A82  YES >>9A8B
  9A84  NO, CLEAR MY TRACE FLAG (PSEUDO TRACE NOW) (BE41)
  9A87  FORCE APPLESOFT TO TRACE FOR MY BENEFIT ONLY
  9A8B  RESTORE A REG AND FALL THRU TO EXIT BI

9A8D ********** SAVE ACTUAL IN/OUT VECTORS **************************

  9A8D  ---
  9A8E  COPY KSWL/H TO VECIN
  9A98  AND CSWL/H TO VECOUT
  9A9A  IN BI GLOBAL PAGE (BE31)
```

```
BASIC Interpreter (BI) -- V1.0.1 -- 1 JAN 84    NEXT OBJECT ADDR: 9AA3
-------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------

9AA3 ********** SET CSWL/KSWL INTERCEPTS ****************************

  9AA3  ---
  9AA4  COPY VDOSIO VECTORS (BE34)
  9AA7  TO CSWL
  9AB1  AND KSWL
  9AB9   EXIT TO CALLER

9ABA ********** INPUT INTERCEPT: MODE = 0 ***************************
         (IMMEDIATE MODE)

  9ABA  IS EXEC FILE ACTIVE? (BE43)
  9ABD  NO >>9AC5
  9ABF  YES, SAVE REGISTERS <9F99>
  9AC2  AND GO READ EXEC FILE FOR INPUT COMMANDS >>9BDE

  9AC5  NO EXEC FILE, RESTORE REAL CSWL/KSWL <9A00>
  9AC8  PROGRAM LOCKED?
  9ACA  YES, DON'T LET HIM INTO IMMEDIATE MODE >>9AFA
  9ACC  NO, READ A KEY FROM KEYBOARD <FD10>
  9ACF  RETURN?
  9AD1  NO, EXIT  >>9AEF
  9AD3  YES, SAVE REGISTERS <9F99>
  9AD6  STORE IT IN LINE BUFFER (0200)
          --> THIS ENTRY CALLED BY EXEC TO PROCESS
              A COMMAND STRING STORED AT $200
  9AD9  GO PROCESS THE COMMAND STRING <A6B4>
  9ADC  CHECK COMMAND NUMBER RETURNED FROM PARSE (BE53)
  9ADF  EXIT BI RIGHT NOW? >>9AEC
  9AE1  NO, COMMAND RETURNED WITH ERROR CODE? >>9B22
  9AE3  NO, RESTORE Y REG (BE40)
  9AE6  RETURN A BACKSPACE TO CALLER OF KEYBOARD
  9AE8  AND A LINE INDEX OF ZERO
  9AEA  EXIT THE BI >>9AEF

  9AEC  RESTORE CALLER'S REGISTERS <9FA3>
  9AEF  AND EXIT BI BY INSTALLING INTERCEPTS >>9A8D

9AF2 ********** SPECIAL ABORT EXIT ***********************************

  9AF2  ERROR=6, "PATH NOT FOUND"
  9AF4  GO SAY SO <BE0C>
  9AFA  SAVE CALLER'S CSWL/KSWL VECTORS <9A8D>
  9AFD  ---
  9AFF  PRINT "PLEASE PRESS SPACE BAR" (BB9A)
  9B0B  FORCE REBOOT ON RESET (03F4)
  9B0E  CHECK KEYBOARD (C000)
  9B13  IS IT A SPACE?
  9B15  NO >>9B0E
  9B17  YES, CLEAR KEYBOARD STROBE (C010)
  9B1A  CLEAR THE SCREEN <FC58>
```

```
BASIC Interpreter (BI) -- V1.0.1 -- 1 JAN 84    NEXT OBJECT ADDR: 9B1D
-----------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-----------------------------------------------------------------

  9B1D  AND CLOSE FILES AND QUIT SYSTEM >>9F8F

9B20 *********** ERROR HANDLER ***************************************

  9B20  ERROR=3, "NO DEVICE CONNECTED"
  9B22  MAIN ENTRY: STORE ERROR CODE (BE0F)
  9B25  AND IN APPLESOFT ONERR
  9B27  CHECK BI STATE (BE42)
  9B2A  MEMORIZE WHETHER IT'S IMMEDIATE MODE
  9B2F  SET A HIGH FILE LEVEL FOR NON-EXEC FILES (BF94)
  9B34  NO ACTIVE READ/WRITE FILES OR PREFIX READ (BE44)
  9B3D  CLOSE ALL OPEN FILES AT OR ABOVE (BEDE)
  9B40  FILE LEVEL = $0F
  9B42  MLI: CLOSE (ALL) <BE70>
  9B45  ERROR? >>9B59
  9B47  WRITE ANY DATA I HAVE BUFFERED <A037>
  9B4A  ERROR? >>9B59
  9B4C  PUT FILE LEVEL BACK TO ZERO
  9B54  NOW FLUSH ALL OPEN FILES
  9B56  MLI: FLUSH (ALL) <BE70>
  9B59  ---
  9B5A  ASSUME MODE WILL BE 4 (DEFERRED)
  9B5C  MEMORIZE WHETHER BASIC ONERR ACTIVE
  9B5E  DEFERRED MODE CURRENTLY? >>9B62
  9B60  NO, STILL IMMEDIATE MODE (MODE=0)
  9B62  ---
  9B63  SET MODE AS DEFINED ABOVE <9FAD>
  9B66  RESTORE BI'S CSWL/KSWL INTERCEPTS <9AA3>
  9B69  GET ERROR CODE (BE0F)
  9B6D  BASIC ONERR ACTIVE? THEN GO HANDLE IT >>9B7F
  9B70  NO, JUST PRINT ERROR MESSAGE <BE0C>
  9B73  CLOSE EXEC FILE IF ONE IS OPEN <B355>
  9B77  DEFERRED MODE? >>9B85
  9B79  IMMED. MODE, PRINT RETURN AND... <9FE2>
  9B7C  WARMSTART APPLESOFT >>D43F

  9B7F  RESTORE STACK FOR BASIC
  9B84   PASS ERROR CODE TO BASIC
  9B85  ---
  9B87  JUMP INTO APPLESOFT ERROR HANDLER >>D865

9B8A *********** RETURN TO IMMED. MODE *******************************

  9B8A  CLEAR APPLESOFT ERRNUM
  9B8E  WILL LOOK FOR "#" FROM APPLESOFT
  9B93  SET NORMAL VIDEO IN APPLESOFT <F273>
  9B96  RESTORE TRUE CSWL/KSWL <9A00>
  9B99  TRY TO WRITE BUFFERED DATA <A02B>
  9B9C  RESET MODE/SET UP BI'S INTERCEPTS <9A17>
  9B9F  RESTORE REGISTERS <9FA3>
  9BA2  GO TO PROCESS IMMED. INPUT REQUEST >>9ABA
```

```
BASIC Interpreter (BI) -- V1.0.1 -- 1 JAN 84    NEXT OBJECT ADDR: 9BA2
-----------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-----------------------------------------------------------------

9BA5 *********** INPUT INTERCEPT: MODE=4 OR 8 ************************

  9BA5  SAVE REGISTERS <9F99>
  9BA8  PREFIX INPUT ACTIVE? (BE46)
  9BAB  NO >>9BB0
  9BAD  YES, GO DO SPECIAL HANDLING >>9D96
  9BB0  ELSE, IS READ FILE ACTIVE? (BE44)
  9BB3  NO >>9BB8
  9BB5  YES, GO DO SPECIAL HANDLING FOR THAT >>9C45
  9BB8  ELSE, IS EXEC FILE ACTIVE? (BE43)
  9BBB  NO >>9BDE
  9BBD  YES, GET PROMPT CHARACTER
  9BBF  IT BETTER NOT BE A "]"
  9BC1  IT IS, RETURN TO IMMEDIATE MODE >>9B8A
  9BC3  ELSE, SET TRUE CSWL/KSWL <9A00>
  9BC6  AND PASS CALLER'S AREG TO REMOVE CURSOR (BE3E)
  9BC9  REMOVE CURSOR AND GET A KEYPRESS <FD10>
  9BCC  BACKSPACE?
  9BCE  NO, EXIT BI >>9BDB
  9BD0  YES, CHECK PROMPT
  9BD2  IF ITS A ">"...
  9BD4  THEN EXIT WITH THE BACKSPACE >>9BD9
  9BD7  ELSE, IF AT START OF LINE, REPROMPT >>9BC6
  9BD9  MIDDLE OF LINE, RETURN A BACKSPACE
  9BDB  EXIT BI TO CALLER >>9A8D

9BDE *********** READ EXEC FILE ***************************************

  9BDE  REMOVE CURSOR FROM SCREEN
  9BE0  CHECK PROMPT CHARACTER
  9BE2  IF ITS A ">"...
  9BE4  DO THINGS DIFFERENTLY >>9C21
  9BE6  CHECK KEYBOARD (C000)
  9BE9  NO KEY READY? >>9BFC
  9BEB  GOT A KEY, IS IT CONTROL-C?
  9BED  NO, IGNORE IT >>9BFC
  9BEF  YES, CLOSE EXEC FILE <B355>
  9BF2  IMMEDIATE MODE? (BE42)
  9BF5  NO >>9C30
  9BF7  YES, CLEAR KEYBOARD STROBE (C010)
  9BFA  AND GO START NEW LINE >>9C30
  9BFC  SET UP FOR EXEC LINE READ <9DB9>
  9BFF  READ A LINE TO $200 <9C9B>
  9C02  ERROR? >>9C29
  9C04  SAVE REGISTERS <9F99>
  9C07  HOP INTO LOOP >>9C0D
  9C09  ---
  9C0A  BACKSCANNING $200 BUFFER (0200)
  9C0D  FORCING THE MSB ON
  9C15  RESTORE TRUE CSWL/KSWL <9A00>
```

```
BASIC Interpreter (BI) -- V1.0.1 -- 1 JAN 84     NEXT OBJECT ADDR: 9C18
-------------------------------------------------------------------------
ADDR     DESCRIPTION/CONTENTS
-------------------------------------------------------------------------

  9C18  GO PROCESS COMMAND LINE <9AD9>
  9C1B  CHECK COMMAND NUMBER (BE53)
  9C1E  IMMEDIATE EXIT? IF NOT, GET NEXT LINE >>9BFC
  9C20  RETURN

          ******** HANDLE EXEC PROMPT > ***********

  9C21  GET SET TO READ EXEC LINE <9DB9>
  9C24  READ SINGLE CHARACTER PER CALL <9C77>
  9C27  NO ERRORS, EXIT TO CALLER NOW >>9C20

          ******** EXEC ERROR RECOVERY ************

  9C29  CLOSE EXEC FILE <B29F>
  9C2C  WAS ERROR "END OF DATA"?
  9C2E  NO, REAL ERROR THEN >>9C42
  9C30  ELSE, OK -- JUST STOP EXECING
  9C32  GET CURSOR HORIZONTAL POSITION
  9C34  IF IN MID LINE, PASS SCREEN CHAR BACK >>9C3D
  9C36  ELSE, CHANGE PROMPT TO "]"
  9C3A  AND RETURN WITH A BACKSPACE
  9C3C  RETURN
  9C3D  GET SCREEN CHARACTER UNDER CURSOR
  9C3F  AND EXIT THRU KSWL TO GET REAL KEYPRESS >>0038
  9C42  REAL ERROR, GO TO BI'S MAIN ERROR HANDLER >>9B22

9C45 *********** INPUT FILE ACTIVE ***********************************

  9C45  GET PROMPT
  9C47  IF ITS A "]"...
  9C4B  THEN RESET TO IMMEDIATE MODE >>9B8A
  9C4E  ELSE, REMOVE CURSOR FROM SCREEN (BE3E)
  9C53  CHECK KEYBOARD (C000)
  9C56  NO KEYPRESS? >>9C60
  9C58  GOT A KEY, IS IT CONTROL-C?
  9C5A  NO, IGNORE IT >>9C60
  9C5C  CLEAR STROBE AND EXIT TO CALLER (C010)
  9C5F  RETURN

  9C60  GET PROMPT AGAIN
  9C62  IS THIS A DIRECTORY FILE? (BE47)
  9C65  YES >>9CC4
  9C67  NO, IS PROMPT = ">"?
  9C69  YES, READ A SINGLE BYTE AT A TIME >>9C71
  9C6B  ELSE, READ ENTIRE LINE <9C96>
  9C6E  ERROR? >>9C42
  9C70  RETURN
```

```
BASIC Interpreter (BI) -- V1.0.1 -- 1 JAN 84     NEXT OBJECT ADDR: 9C70
-------------------------------------------------------------------------
ADDR     DESCRIPTION/CONTENTS
-------------------------------------------------------------------------

  9C71  READ SINGLE BYTE FROM INPUT FILE <9C77>
  9C74  ERROR? >>9C42
  9C76  RETURN

9C77 *********** READ NEXT BYTE OF FILE ******************************

  9C77  SAVE CURRENT READ/WRITE COUNT (BED9)
  9C7A  IN L KEYWORD VALUE (BE5F)
  9C7F  SET UP TO READ ONE BYTE (BED9)
  9C84  MLI: READ <BE70>
  9C87  ERROR? >>9C95
  9C89  PUT COUNT BACK TO MAXIMUM AGAIN (BE5F)
  9C8F  GET FIRST CHARACTER ON $200 LINE (BED7)
  9C92  AND RETURN THAT TO CALLER (0200)
  9C95  RETURN

9C96 *********** READ NEXT LINE OF FILE ******************************

  9C96  REMOVE CURSOR FROM SCREEN (BE3E)
  9C9B  ---
  9C9D  MLI: READ <BE70>
  9CA0  ERROR? >>9C95
  9CA2  GET LENGTH ACTUALLY TRANSMITTED (BEDB)
  9CA5  NOTHING? >>9CBD
  9CA8  GOT SOMETHING, FIND END OF DATA (BED7)
  9CAC  FETCH LAST BYTE OF LINE (01FF)
  9CB1  IS IT A RETURN CHARACTER?
  9CB3  NO, LEAVE LINE ALONE >>9CBD
  9CB5  YES, WAS L KEYWORD GIVEN? (BE57)
  9CBA  YES, LEAVE IT BE >>9CBD
  9CBC   ELSE, CHOP OFF THE RETURN ITSELF
  9CBD  AND EXIT WITH A RETURN
  9CBF  RESTORING Y REG AS YOU GO (BE40)
  9CC3  RETURN

9CC4 *********** READING DIR FILE ***********************************

  9CC4  ">" PROMPT?
  9CC6  YES, EXIT RIGHT NOW >>9CBD
  9CC8  ELSE, REMOVE CURSOR FROM SCREEN (BE3E)
  9CCD  SET 80 COLUMNS
  9CD4  MLI: GET MARK <BE70>
  9CD7  ERROR? >>9D4E
  9CD9  ARE WE AT BEGINNING OF THIS FILE? (BEC8)
  9CDF  NO, CONTINUE >>9D0E
  9CE1  YES, CAT FLAG = 2
  9CE6  READ DIRECTORY HEADER <B1B7>
  9CE9  ERROR? >>9D4E
  9CEB  REF NUM TIMES 32 (BED6)
  9CF6  SET THE L VALUE OF THIS DIR FILE IN (BCFF)
```

```
BASIC Interpreter (BI) -- V1.0.1 -- 1 JAN 84    NEXT OBJECT ADDR: 9CF9
------------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
------------------------------------------------------------------------

 9CF9  THE OPEN FILE LIST TO THE ENTRY LENGTH (BCB8)
 9CFC  AND THE NUMBER OF ENTRIES PER BLOCK (BD00)

        ******** FORMAT DIRECTORY NAME **********

 9CFF  GO FORMAT NAME OF DIRECTORY <B112>
 9D02  STORE THE LENGTH OF LINE AT $200
 9D07  PUT A RETURN CHAR AT END OF LINE
 9D0C   AND EXIT TO CALLER
 9D0D  RETURN

 9D0E  GET CAT FLAG (BE4F)
 9D11  IF ZERO, GO PROCESS INDIVIDUAL ENTRIES >>9D51
 9D13  IF MINUS, GO DO SUMMARY LINE OR EXIT >>9D28
 9D15  POSITIVE, ASSUME NULL LINE WANTED
 9D17  DROP CAT FLAG BY ONE (BE4F)
 9D1A  IF ZERO, JUST GO PRINT A BLANK LINE >>9D02

        ******** FORMAT TITLE LINE **************

 9D1C  ELSE, BLANK OUT $200 AND  <A6A9>
 9D21  UNPACK "NAME TYPE BLOCKS ETC... <9FE7>

 9D24  LINE LENGTH IS 80
 9D26  GO RETURN IT TO CALLER >>9D02

        ******** FORMAT SUMMARY LINE ************

 9D28  DO SUMMARY LINE?
 9D2A  NO, JUST EXIT (ALL DONE) >>9D4B
 9D2C  YES, DROP CAT FLAG SO EXIT NEXT TIME (BE4F)
 9D31  CLEAR READ/WRITE COUNT (BED9)
 9D39  MLI: READ <BE70>
 9D3C  FORMAT BLOCKS FREE AND INUSE SUMMARY LINE <B141>
 9D40  GET REF NUM (BED6)
 9D43  AND COPY TO GET/SET LIST (BEC7)
 9D47  NO ERRORS, EXIT >>9D24
 9D49  ERROR, JUMP TO BI ERROR EXIT >>9D4E
 9D4B  "END OF DATA" ERROR
 9D4E  GO TO BI ERROR EXIT >>9B22

        ******** FORMAT FILE/DIR ENTRIES ********

 9D51  SET DIR ENTRY NUM COUNTER TO -1
 9D56  GET REF NUM (BED6)
 9D59  *32
 9D5E  USE AS INDEX TO GET ENTRY LENGTH (BCFF)
 9D64  AND ENTRIES PER BLOCK FROM OPEN FILE LIST (BD00)
 9D6A  POSITION ON EVEN BLOCK BOUNDARY (BEC9)
 9D70  AND GET SECTOR OFFSET (BEC8)
 9D74  SKIP FILE/DIR ENTRIES UNTIL POSITIONED TO (BCBB)
```

```
BASIC Interpreter (BI) -- V1.0.1 -- 1 JAN 84    NEXT OBJECT ADDR: 9D77
------------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
------------------------------------------------------------------------

 9D77  CURRENT POSITION IN THIS BLOCK (BCB7)
 9D7F  READ NEXT DIR ENTRY FROM FILE <B22B>
 9D82  NO ERROR? >>9D90
 9D84  ERROR, IF RANGE ERROR...
 9D86  NO, TRUE ERROR >>9D4E
 9D88  RANGE ERROR, READY FOR SUMMARY LINE NEXT (BE4F)
 9D8D  RETURN A BLANK LINE THIS TIME >>9D02

 9D90  FORMAT FILE/DIR ENTRY INTO $201  <A501>
 9D93  AND RETURN IT TO CALLER >>9D24

9D96 ********** PREFIX INPUT ACTIVE *********************************

 9D96  PROMPT = "]"?
 9D98  NO, ALL IS WELL >>9D9D
 9D9A  YES, RETURN TO IMMEDIATE MODE NOW >>9B8A
 9D9D  REMOVE CURSOR FROM SCREEN (BE3E)
 9DA4  PREFIX NO LONGER ACTIVE AFTER THIS (BE46)
 9DAA  COPY PATHNAME BUFFER (PREFIX) (BCBC)
 9DAD  TO $200 (01FF)
 9DB3  RETURN WITH IT TO BASIC (BCBC)
 9DB8  RETURN

9DB9 ********** SETUP TO READ LINE FROM EXEC ************************

 9DB9  SET READ REF NUM FOR EXEC FILE (BCA3)
 9DBF  READ TO $200
 9DC4  FOR $EF BYTES OF LENGTH
 9DC9  (OR UNTIL A RETURN CHAR)
 9DD1  RETURN

9DD2 ********** OUTPUT INTERCEPT: MODE = C ***************************
        (LOOK FOR CONTROL-D)

 9DD2  SAVE REGISTERS <9F99>
 9DD5  PRINTING A CONTROL-D?
 9DD7  NO >>9DF0
 9DD9  YES, WRITE OUT ANY BUFFERED DATA <A02B>
 9DDC  NOTHING IN COMMAND LINE (BE4B)
 9DDF  READ FILE INACTIVE (BE44)
 9DE2  WRITE FILE INACTIVE (BE45)
 9DE5  PREFIX READ INACTIVE (BE46)
 9DEA  SET MODE = 8 FROM NOW ON <9FAD>
 9DED  RESTORE REGS AND EXIT >>9FA3

 9DF0  GOT A CONTROL-D...
 9DF2  SET MODE = 4 FROM NOW ON <9FAD>
 9DF5  RESTORE REGISTERS <9FA3>
 9DF8  OUTPUT CHARACTER AND EXIT >>B84B
```

```
BASIC Interpreter (BI) -- V1.0.1 -- 1 JAN 84    NEXT OBJECT ADDR: 9DF8
-------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------

9DFB ********** OUTPUT INTERCEPT: MODE = 8 *************************
      ,   (ASSEMBLE COMMAND LINE)

  9DFB  SAVE REGISTERS <9F99>
  9E01  SAVE CHAR IN COMMAND LINE (0200)
  9E04  WAS IT A RETURN?
  9E06  YES, READY TO RDLL >>9E16
  9E08  ND, BUMP CHARACTER COUNTER (BE4B)
  9E0B  AND EXIT TO CALLER >>9E12
  9E0D  OOPS! LINE TOO LONG
  9E0F  "SYNTAX ERROR" >>9B22
  9E12  ELSE, RESTORE X REG AND EXIT (BE3F)
  9E15  RETURN

  9E16  ---
  9E18  NULL LINE? >>9E25
  9E1A  NO, PUT BACK TRUE CSWL/KSWL <9A00>
  9E1D  SYNTAX SCAN CMD LINE <A6B4>
  9E20  ERROR? >>9E0F
  9E22  NO, PUT BACK BI'S INTERCEPTS <9A8D>
  9E25  ---
  9E27  MODE = 4 NOW <9FAD>
  9E2A  RESTORE REGS AND EXIT >>9FA3

9E2D ********** WRITE BUFFERED CHARACTER ***************************

  9E2D  SAVE Y REG (BE40)
  9E30  CHECK PROMPT
  9E32  CHECK TO SEE IF WE ARE IN "IF", >>9E40
  9E35  "PRINT", "LIST", OR "CALL" STATEMENTS >>9E40
  9E38  OF AN APPLESOFT PROGRAM >>9E40
  9E3A  IF NOT, EXIT TO CALLER... (BE40)
  9E3D  WITH CHARACTER ECHOED TO SCREEN >>9A74

  9E40  GET INDEX TO TEMPORARILY BUFFERED CHARS (BE4A)
  9E45  STORE INTO BUFFER JUST ABOVE HIMEM
  9E4A  BUMP INDEX (BE4A)
  9E4D  OK >>9E5A
  9E4F  BUFFER FULL, SAVE REGISTERS <9F99>
  9E52  WRITE BUFFER OUT TO DISK <A025>
  9E55  ERROR? >>9E0F
  9E57  RESTORE REGISTERS <9FA3>
  9E5A   AND EXIT ANYWAY

9E5B ********** OUTPUT INTERCEPT: MODE = 4 *************************
          (INITIAL ENTRY FOR A RUNNING PROGRAM)
          (FLUSH OUT NON COMMAND LINES)
```

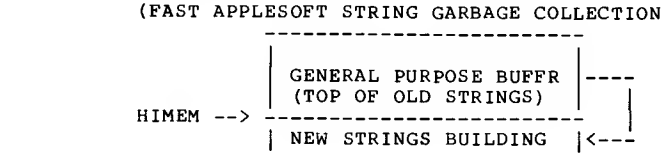
```
BASIC Interpreter (BI) -- V1.0.1 -- 1 JAN 84    NEXT OBJECT ADDR: 9E5B
-------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------

  9E5B  PRINTING A "#"? (9F98)
  9E5E  ND >>9E78
  9E60  YES, SAVE X REGISTER (BE3F)
  9E64  RETURN ADDR IS IN APPLESOFT... (0103)
  9E67  TRACE ROUTINE...
  9E6B  AT $D812? (0104)
  9E70  YES >>9EE5
  9E72  NO, RESTORE REGISTERS (9F98)
  9E78  IS WRITE FILE ACTIVE?  (BE45)
  9E7B  NOPE >>9E9B
  9E7D  YES, PRINTING A "]"?
  9E7F  NO >>9E85
  9E81  YES, SAME AS PROMPT CHARACTER?
  9E83  YES >>9EB5
  9E85  NO, PRINTING A RETURN CHAR?
  9E87  NO >>9E2D
  9E89  YES, GET PROMPT
  9E8F  DOES IT INDICATE RECURSION? >>9E2D
  9E91  YES, WRITE BUFFER OUT <A02B>
  9E94  OUTPUT FILE INACTIVE NOW (BE45)
  9E99  EXIT WITH RETURN CHAR >>9ECE

  9E9B  ---
  9E9C  INPUT FILE ACTIVE? (BE44)
  9EA2  NO >>9EAC
  9EA4  YES, CHECK PROMPT
  9EA6  OR IN $04
  9EA8  CONTROL-D?
  9EAA  YES >>9ED1
  9EAC  ---
  9EAD  NO, HOW BOUT "]"?
  9EAF  NO, EXIT WITH ECHO THEN >>9ECE
  9EB1  YES, IS THIS THE PROMPT CHAR?
  9EB3  NO, EXIT WITH ECHO >>9ECE
  9EB5  YES, SAVE REGISTERS <9F99>
  9EB8  CHECK OPEN FILE COUNT (BE4D)
  9EBB  NONE OPEN? >>9ECB
  9EBD  SOME OPEN, WRITE BUFFER OUT <A02B>
  9EC0  INDICATE WRITE FILE INACTIVE NOW (BE45)
  9EC3  SET TRUE CSWL/KSWL <9A00>
  9EC8  PRINT "FILE(S) STILL OPEN" <BE0C>
  9ECB  RESTORE REGS <9FA3>
  9ECE  AND ECHO EXIT >>9A74

  9ED1  ---
  9ED2  CHAR IS A RETURN?
  9ED4  NO >>9ED9
  9ED6  YES, SAME AS LAST CHAR OUTPUT? (BE4C)
  9ED9  (SAVE IT FOR THIS TEST NEXT TIME) (BE4C)
  9EDC  NOT SAME, NO PROBLEM THEN >>9EE0
```

```
BASIC Interpreter (BI) -- V1.0.1 -- 1 JAN 84    NEXT OBJECT ADDR: 9EDE
-------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------

  9EDE  SAME, MARK PROMPT FOR RECURSION
  9EE0  RETURN

9EE1 *********** APPLESOFT TRACE INTERCEPT ***************************
         (CONTROL PASSES HERE FOR EVERY STATEMENT)
         (EXECUTED WHILE PRODOS IS ACTIVE)

  9EE1  BUMP APPLESOFT LINE POINTER
  9EE5  ---
  9EE9  MARK PROMPT FOR RECURSION
  9EEB  JUST IN CASE WE DIE IN HERE
  9EED  RESTORE APPLESOFT'S STACK
  9EF0  DOES BI KNOW WE ARE TRACING? (BE41)
  9EF3  YES, REAL LIVE TRACE THEN >>9F68

  9EF5  ELSE, PICK UP NEXT TOKEN ON LINE
  9EF9  IS IT A TOKEN? >>9F20
  9EFB  OR END OF LINE? >>9F1D
  9EFD  NEITHER, DECREMENT STRING SPACE CTR (BE49)
  9F00  OK >>9F1B
  9F02  COMPUTE SIZE OF FREESPACE IN PAGES
  9F06  AT LEAST 3 PAGES AVAILABLE?
  9F08  YES >>9F14
  9F0A  NO, WRITE BUFFERED DATA <A02B>
  9F0D  AND THEN GARBAGE COLLECT <A07B>
  9F12  COMPUTE FREE SPACE NOW
  9F14  AND SAVE IN STRING SPACE CTR (BE49)
  9F19  GET NEXT TOKEN
  9F1B  ---
  9F1D  JUMP BACK INTO APPLESOFT TO EXECUTE IT >>D820
  9F20  STORE TOKEN IN PROMPT
  9F23  LOOK UP TOKEN IN BI'S TOKEN TABLE (B7F3)
  9F26  ITS NOT ONE BI IS INTERESTED IN >>9F1D
  9F28  IT IS INTERESTING, CHANGE BRANCH (9F2C)
  9F2B  AND JUMP TO ONE OF THE FOLLOWING: >>9F2D

  9F2D  IF OR PRINT: PROMPT = 0
  9F2F  CLEAR OUT LAST CHAR SAVEAREA (BE4C)
  9F32  GO TO MODE = C NEXT TIME THRU (B85D)
  9F35  (BEGIN LOOKING FOR COMMANDS) (BE38)
  9F3E  NOW GO PROCESS THE IF OR PRINT >>9F5D

  9F40  LIST: PROMPT = 1
  9F42  (DON'T LOOK FOR COMMANDS NOW)
  9F44  GO DO IT >>9F5D

  9F46  CALL: PROMPT = 2
  9F48  (DON'T LOOK FOR COMMANDS NOW)
  9F4A  GO DO IT >>9F5D
```

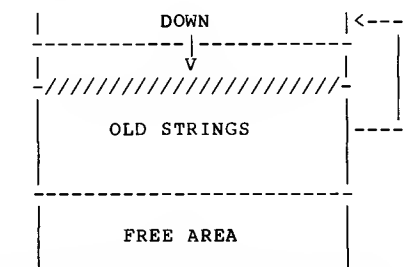
```
BASIC Interpreter (BI) -- V1.0.1 -- 1 JAN 84    NEXT OBJECT ADDR: 9F4A
-------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------

  9F4C   LET: DECREMENT STRING CTR
  9F4D  AND GO BACK FOR NEXT TOKEN >>9EFD

  9F50  TRACE: TURN TRACE ON (BE41)
  9F53  THEN CONTINUE BELOW >>9F59

  9F55  NOTRACE: DROP INTO BACKGROUND TRACE (BE41)
  9F58   CHANGE TOKEN TO "TRACE"
  9F59  FORCE ON APPLESOFT TRACE
  9F5D  ---
  9F5E  GO BACK TO APPLESOFT TO PERFORM IT >>D820

  9F61  RESUME: CLEAR ONERR CODE
  9F66  GO TO APPLESOFT TO PROCESS IT >>9F1B

          ******** REAL TRACE ACTIVE **************

  9F68  RESTORE TRUE CSWL/KSWL <9A00>
  9F6D  PRINT "#" <FDED>
  9F74  USE APPLESOFT TO PRINT CURRENT LINE NO. <ED24>
  9F79  PRINT A BLANK SPACE <FDED>
  9F7C  PUT BI'S CSWL/KSWL INTERCEPTS BACK <9A8D>
  9F80  THEN GO BACK AND HANDLE AS USUAL >>9EF5

  9F83  LOOKING FOR A LOWER CASE "c"
  9F87  LOOKING FOR A "#"
  9F89  STORE CHAR TO SEARCH FOR (9F98)
  9F8D  BRANCH BACK INTO APPLESOFT >>9F1B

9F8F *********** EXIT SYSTEM VECTOR **********************************

  9F8F  CLOSE ALL OPEN FILES <B54C>
  9F94  MLI: QUIT <BE70>

  9F98  "#" CHARACTER (ASOFT TRACE CHAR)

9F99 *********** SAVE CALLERS REGISTERS ******************************

  9F99  SAVE A,X AND Y REGS (BE3E)
  9FA2  RETURN

9FA3 *********** RESTORE CALLERS REGISTERS ****************************

  9FA3  RESTORE A,X AND Y REGS (BE3E)
  9FAC  RETURN
```

```
BASIC Interpreter (BI) -- V1.0.1 -- 1 JAN 84    NEXT OBJECT ADDR: 9FAC
-------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------

9FAD ********** SET MODE AND CSWL/KSWL ******************************

  9FAD' STORE "STATE" MODE FROM X REGISTER (BE42)
  9FB2  COPY PROPER CSWL/KSWL VALUES TO REDIRECT... (B851)
  9FB5  VECTOR DEPENDING ON CURRENT MODE (BE38)
  9FBE  RETURN

9FBF ********** PRINTERR: PRINT ERROR MSG ***************************

  9FBF  ---
  9FC0  GET INDEX INTO PACKED MESSAGE TEXTS (BA65)
  9FC3  UNPACK MESSAGE INTO $201 <9FE7>
  9FC9  SAVE THE LENGTH (BCB6)
  9FCC  SKIP A LINE <9FE2>
  9FD1  PRINT A BELL <9FE4>
  9FD4  ---
  9FD6  PRINT CONTENTS OF $201 MSG BUFFER (0201)
  9FE2  PRINT A RETURN CHARACTER
  9FE4  AND EXIT >>FDED

9FE7 ********** UNPACK ERROR MESSAGE *********************************

  9FE7  NOTHING IN BUFFER AT FIRST
  9FED  GET A NIBBLE FROM PACKED MSG <A009>
  9FF0  NON-ZERO, COMMON CHARACTER >>9FF7
  9FF2  IF ZERO, GET NEXT NIBBLE <A009>
  9FF5  AND CONVERT TO UNCOMMON CHAR INDEX
  9FF7  ---
  9FF8  GET THE LETTER THIS NIBBLE REPRESENTS (BA7A)
  9FFB  ZERO? THEN END OF MESSAGE >>A008
  9FFD  GET INDEX INTO OUTPUT BUFFER (BE4B)
  A000  AND STORE THE CHARACTER THERE (0201)
  A003  BUMP INDEX (BE4B)
  A006  AND CONTINUE >>9FED
  A008  RETURN

A009 ********** UNPACK MESSAGE BYTE *********************************

  A009  GET NEXT MSG BYTE (BA9A)
  A00C  WORKING ON SECOND NIBBLE? >>A020
  A00E  NO, TAB INDICATOR? >>A016
  A010  NO, ISOLATE HIGH NIBBLE
  A014   NEXT TIME GET LOW NIBBLE
  A015  RETURN

  A016  ---
  A017  GET TAB POSITION (BA9A)
  A01A  AND BUMP OUTPUT PTR ACCORDINGLY (BE4B)
  A01E  THEN GO BACK FOR NEXT NIBBLE >>A009
```

```
BASIC Interpreter (BI) -- V1.0.1 -- 1 JAN 84    NEXT OBJECT ADDR: A01E
-------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------

  A020   BUMP BYTE PTR FOR NEXT TIME
  A021  ISOLATE LOW NIBBLE
  A023   NEXT TIME GET HIGH NIBBLE
  A024  RETURN

A025 ********** WRITE ONE BUFFERED BYTE ******************************

  A025  SET UP COUNT OF 0001
  A029  AND JUMP INTO ROUTINE BELOW >>A03E

A02B ********** WRITE BUFFERED DATA/TEST ERROR ***********************

  A02B  WRITE BUFFERED DATA <A037>
  A02E  OK? THEN EXIT >>A053
  A031   ERROR, POP OUT OF THIS SUBROUTINE
  A034  AND GO TO ERROR HANDLER >>9B22

A037 ********** WRITE ALL BUFFERED DATA ******************************

  A037  ---
  A039  GET BUFFERED DATA COUNT (BE4A)
  A03C  NONE BUFFERED? >>A052
  A03E  STORE BUFFERED DATA COUNT IN RW PARMS (BED9)
  A046  MLI: WRITE <BE70>
  A04C  NOTHING BUFFERED NOW, COUNT=0 (BE4A)
  A050  ERROR? >>A053
  A052   NO, EXIT
  A053  RETURN

A054 ********** SPECIAL GARBAGE COLLECT ******************************
          (PULL OUT STRING CONSTANTS ALSO)

  A054  DO GARBAGE COLLECTION NORMALLY FIRST <A07B>
  A057  ERROR? >>A07A
  A05B  START OF STRING AREA = PROGRAM START PTR (BC84)
  A063  USE GENERAL PURPOSE BUFFER (ABOVE HIMEM)
  A065  FOR A GARBAGE COLLECT WORKAREA (BC7D)
  A06A  IT IS 3+1 PAGES IN LENGTH (BC7E)
  A06F  END OF STRING AREA IS AT END OF FREEAREA (BC86)
  A077  GO COLLECT CONSTANT STRINGS NOW <A0C2>
  A07A   THEN EXIT

A07B ********** "FRE" COMMAND ****************************************
          (FAST APPLESOFT STRING GARBAGE COLLECTION)
```



```
BASIC Interpreter (BI) -- V1.0.1 -- 1 JAN 84    NEXT OBJECT ADDR: A07B
-------------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------------
```




```
        TOP PART OF OLD STRINGS IS SAVED IN THE
        GENERAL PURPOSE BUFFER OR IN THE FREE
        AREA (WHICHEVER IS LARGER) AND A NEW
        COPY OF THE STRINGS IS BUILT JUST BELOW
        HIMEM.

  A07B  STRING AREA START IS ON PAGE BOUNDARY
  A082  ASSUME 4 PAGE WORKAREA (BC7E)
  A087  IN GENERAL PURPOSE BUFFER ABOVE HIMEM (BC7D)
  A08C  STRING START PTR IS START OF STRING AREA (BC84)
  A090  COMPUTE NUMBER OF FREE PAGES
  A092  AT LEAST 7?
  A094  IF NOT, USE G.P. WORKAREA INSTEAD >>A0B0
  A096  DON'T USE ALL OF FREE AREA (LEAVE $300)
  A098  NEW WORKAREA SIZE IS FREE AREA SIZE-$300 (BC7E)
  A09D  SET PTR TO WORKAREA AT FIRST FREE PAGE
  A0A4  COMPUTE NUMBER OF STRING PAGES
  A0A8  USE SMALLER OF STRING PAGES OR WORKAREA SIZE (BC7E)
  A0AD  AS NEW WORKAREA SIZE (BC7E)
  A0B0  END OF STRING AREA IS HIMEM
  A0BA  JUMP TO NEXT INSTRUCTION >>A0BD
  A0BD  STRING START LSB IS HIMEM INITIALLY (BC85)
  A0C2   RECORD WHETHER LAST PAGE IS PARTIAL
  A0C6  STRING START MSB IS HIMEM INITIALLY (BC86)
  A0CB  ADJUST LORANGE AND HIRANGE MSB'S
  A0CD  FOR PARTIAL PAGES AT EITHER END, (BC7F)
  A0D0  SETTING THEM AT HIMEM FOR NOW.
  A0D9  SET UP ARRAY END MSB +1 FOR COMPARES (BC82)
  A0DC  $3E/$3F --> FIRST VARIABLE (LESS 7 BYTES)
  A0DE  (EACH VARIABLE IS 7 BYTES)
  A0E8  SET UP ARRAY START LSB FOR COMPARES
  A0ED  GET LORANGE VALUE (BC7F)
  A0F0  PRIOR TO STRING AREA? (BC84)
  A0F3  YES, THEN DONE! >>A133
  A0F5  ELSE, DROP LORANGE BY WORKAREA SIZE (BC7E)
  A0F8  AND SAVE THIS VALUE (BC7C)
  A0FB  NOW DROP IT ALSO BY THE DISTANCE BETWEEN
  A0FD  ...THE OLD LORANGE AND THE STRING START PTR (BC7F)
  A107  USE THE LOWER OF THE TWO VALUES (BC7C)
```

```
BASIC Interpreter (BI) -- V1.0.1 -- 1 JAN 84    NEXT OBJECT ADDR: A10C
-------------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------------

  A10C  TO PRODUCE THE MAXIMUM SIZED RANGE (BC7C)
  A10F  IS THIS BELOW THE BOTTOM OF THE STRINGS? (BC84)
  A112  NO >>A119
  A114  YES, USE THE BOTTOM POINTER INSTEAD (BC84)
  A117  (ADJUSTING FOR PARTIAL PAGE)
  A119  STORE FINAL LORANGE VALUE (BC7F)
  A11C  COPY SOME PAGES BELOW HIRANGE TO WORKAREA <A1D2>
  A11F  (TO MAKE ROOM FOR NEW STRINGS)
  A121  COLLECT SIMPLE STRING VARS FOR THIS RANGE <A134>
  A124  ERROR? >>A131
  A126  THEN COLLECT STRING ARRAYS <A16A>
  A129  NEW HIRANGE = OLD LORANGE (BC7F)
  A12F  CONTINUE LOOPING >>A0DC

  A131  IF ERROR, "RAM TOO LARGE"
  A133   EXIT TO CALLER

A134 ********** COLLECT SIMPLE STRINGS ******************************

  A134  ---
  A135  ADD 7 BYTES TO $3E/$3F PTR FOR NEXT VAR
  A13F  PTR AT ARRAYS NOW?
  A145  IF SO, WE ARE DONE >>A168
  A147  IS THIS A STRING VARIABLE?
  A14E  NO >>A134
  A150  MAKE ABSOLUTELY SURE
  A154  GET MSB OF STRING POINTER
  A158  IS IT WITHIN MY RANGE? (BC7F)
  A15B  NO >>A135
  A160  NO >>A134
  A162  YES, PULL IT OUT AND TACK IT TO HIMEM <A1F5>
  A165  ALL WENT WELL, GET NEXT VARIABLE >>A135
  A167   IF ERROR, EXIT NOW

  A168   NORMAL EXIT TO CALLER
  A169  RETURN

A16A ********** COLLECT STRING ARRAYS ********************************

  A16A  FIND THE NEXT ARRAY <A199>
  A16D  NO MORE? >>A168
  A16F  GOT ONE, GET MSB OF ITS STRING PTR
  A173  WITHIN MY RANGE? (BC7F)
  A176  NO >>A183
  A17B  NO >>A183
  A17D  YES, PULL IT OUT AND TACK IT TO HIMEM <A1F5>
  A180  AND CONTINUE WITH NEXT ARRAY ELEMENT >>A184
  A182   ERROR EXIT
```

```
BASIC Interpreter (BI) -- V1.0.1 -- 1 JAN 84    NEXT DBJECT ADDR: A182
-------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------

  A183  ---
  A184  BUMP PDINTER TO NEXT ARRAY MEMBER
  A18E' POINTER NOW AT NEXT ARRAY? (BC81)
  A191  ND, DO THIS ELEMENT >>A16F
  A195  ND >>A16F
  A197  YES, SET UP TO PRDCESS THAT ONE THEN >>A16A

A199 *********** FIND NEXT STRING ARRAY ******************************

  A199  ---
  A19A  $3E --> ARRAY VARIABLES (BC81)
  A1A1  AT END OF ARRAY VARS
  A1A3  NO, CDNTINUE >>A1A9
  A1A7  YES, DUT (CARRY SET, NO MDRE ARRAYS) >>A1D1

  A1A9  POINT TD ARRAY FDLLDWING THIS (LSB AND...)
  A1B3  MSB TD X REGISTER
  A1BA  CHECK TYPE OF VARIABLE
  A1BF  SKIP INTEGER AND REAL ARRAYS >>A199
  A1C3  GET NUMBER OF DIMENSIDNS
  A1C5  *2 TD SKIP SIZES
  A1C6  +5 TO SKIP FIXED STUFF AT BEGINNING
  A1CA  PDINT TD FIRST ARRAY MEMBER
  A1CE  READY TD ROLL, $3E POINTS TD IT
  A1D1  RETURN

A1D2 *********** COPY PAGES TO WDRKAREA ******************************
          TO MAKE ROOM FDR NEW STRINGS BEING MDVED
          TO HIMEM, CDPY SOME STRING PAGES FROM OLD
          STRING AREA TO THE WORKAREA TO PROTECT THEM.

  A1D2  $3A/$3B --> FIRST PAGE TD SAVE (BC7C)
  A1D7  $3C/$3D --> WORKAREA (BC7D)
  A1E2  CDPY N+1 PAGES (SIZE OF WORKAREA) (BC7E)
  A1E6  ---
  A1F4   EXIT WHEN FINISHED

A1F5 *********** PULL STRING DUT **************************************
          TACK STRING JUST UNDER HIMEM AT CURRENT
          STRING START PDINTER.

  A1F5  IS STRING BELOW SAVED AREA? (BC7C)
  A1F8  YES, ITS STILL THERE THEN >>A201
  A1FA  ELSE, PDINT TO SAVED STRING IN WORKAREA (BC7C)
  A201  $3A/$3B --> STRING
  A20C  DROP STRING START PTR BY LEN DF THIS STRING
  A211  UPDATE STRING'S LSB IN VARIABLE PTR
  A215  FIX UP MSB DF STRING START PTR ALSD
  A21A  AND OF VARIABLE PTR
  A21E  IS THIS A NULL LENGTH STRING?
```

```
BASIC Interpreter (BI) -- V1.0.1 -- 1 JAN 84    NEXT OBJECT ADDR: A220
-------------------------------------------------------------------
ADDR    DESCRIPTION/CDNTENTS
-------------------------------------------------------------------

  A220  YES, ND MOVE TO DO >>A22B
  A223  ---
  A224  ELSE, CDPY STRING OUT
  A22B  ---
  A22C  DUT OF FREESPACE? (BC82)
  A231   RETURN TO CALLER WITH INDICATION

A232 *********** ALLDCATE BUFFER **************************************

  A232  NEED 4 PAGES

          ******** GENERAL PURPOSE ALLOCATE *******

  A234  STORE THAT (BBB6)
  A237  GD GARBAGE CDLLECT TO GET SPACE <A07B>
  A23A  ERRDR? >>A287
  A23E  HOW MANY FREE PAGES ARE THERE?
  A240  ARE THERE ENOUGH? (BBB6)
  A243  IF NOT, "RAM TOO LARGE" MSG
  A245  TOD FEW... >>A287
  A247  GDT ENOUGH, $3A-->TOP OF FREESPACE
  A24E  AND $3C-->NEW TDP AFTER ALLOCATION
  A258  COMPUTE LENGTH DF STRINGS FOR COPY
  A266  COPY STRINGS DOWN "N" PAGES IN MEMORY <A396>
  A26C  SUBTRACT "N" FROM STRING ADDRESS MSB'S (BBB6)
  A272  ADJUST ALL POINTERS IN SIMPLE & ARRAY VARS <A3DA>
  A277  DLD HIMEM BECOMES BUFF ADDR HIGH WATER MARK (BBB8)
  A27E  NEW HIMEM IS "N" PAGES LDWER
  A283  FIND PAGE JUST BEYDND A FILE BUFFER (BC88)
  A286  RETURN
  A287  ---
  A288  RETURN

A289 *********** FREE BUFFER ******************************************

  A289  GARBAGE COLLECT STRINGS <A07B>
  A28C  ERROR? >>A2D4
  A292  PUT HIMEM-$100 INTD $3A/3B
  A296  AND HIMEM+$400 INTD $3C/3D
  A29C  (COPY LSB'S)
  A2A3  BC92 = LENGTH OF STRINGS (BC92)
  A2AD  COPY STRINGS UP 4 PAGES <A3BA>
  A2B2  PREPARE TD ADJUST THEM BY $400 (BC87)
  A2B8  NEW HIMEM+$400
  A2BA  ADJUST ALL STRING ADDRS UP BY $400 <A3DA>
  A2C0  ARE WE FREEING BOTTOM-MOST BUFFER?
  A2C2  YES, DDNE! >>A2EE
  A2C5  CHECK DPEN FILE CDUNT (BE4D)
  A2C8  NONE DPEN? (HOW CAN THAT BE?) >>A2D4
  A2CA  WHICH FILE'S BUFFER IS NEXT TD HIMEM?
  A2CF  SEARCH UNTIL IT IS FDUND... >>A2D5
```

```
BASIC Interpreter (BI) -- V1.0.1 -- 1 JAN 84    NEXT OBJECT ADDR: A2D4
-------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------

  A2D4   RETURN IF NO FILE IS USING THIS BUFFER
  A2D5   ---
  A2D6   GIVE THAT FILE THE BUFFER PASSED TO US (BEC9)
  A2D9   (SURE HOPE THAT FILE WAS FLUSHED!) (BC93)
  A2E4   PASS FILE REF NUM TO MLI (BEC7)
  A2E9   MLI: SET NEW BUFFER <BE70>
  A2EC   ERROR? >>A2D4
  A2EE   ---
  A2EF   RETURN

A2F0 *********** GETBUFR: GET A BUFFER *******************************
          THIS ROUTINE IS CALLED THROUGH AN EXTERNAL
          ENTRY POINT IN THE GLOBAL PAGE.  IT ALLO-
          CATES A FIXED LOCATION BUFFER BETWEEN THE
          BI AND ITS BUFFERS.

  A2F0   ALLOCATE A BUFFER OF ANY SIZE (A=PAGES) <A234>
  A2F3   ERROR? >>A33A
  A2F8   FIND FIRST PAGE OF BUFFER (BBB9)
  A2FF   GET FILE OPEN COUNT (BE4D)
  A302   NONE OPEN? >>A325
  A304   BUMP BUFFER PAGE PTR BY $400 (BBB8)
  A308   TO POINT TO PREVIOUSLY ALLOCATED
  A30A   BUFFER. (BBB8)
  A30D   FIND OPEN FILE WITH THIS BUFFER (BC93)
  A312   GOT IT, (BEC9)
  A315   SET FILE BUFFER REAL LOW IN MEMORY <A38D>
  A318   THEN SET IT TO NEW BUFFER LOCATION <A2D6>
  A31B   BELOW ALL OTHERS (BEC9)
  A322    DO THIS FOR EACH OPEN FILE...
  A323   THEREBY INSERTING A BLANK BUFFER >>A30D
  A328   IS EXEC FILE ACTIVE? (BE43)
  A32B   NO, DONE >>A33A
  A32D   YES,
  A32F   MOVE EXEC BUFFER DOWN ALSO <A38D>
  A338   AND BUMP UP ABOVE IT
  A33A    EXIT TO CALLER
  A33B   RETURN

A33C *********** FREEBUFR: FREE BUFFER *******************************
          THIS ROUTINE IS CALLED THROUGH AN EXTERNAL
          ENTRY POINT IN THE GLOBAL PAGE.  IT FREES
          A FIXED LOCATION BUFFER PREVIOUSLY ALLO-
          CATED BY GETBUFR.

  A33C   GET COUNT OF OPEN FILES (BE4D)
  A340   INDEX THIS BY 4 PAGES PER FILE
  A341   ADD TO HIMEM MSB
  A343   SAVE THIS AS TOP OF BUFFERS (BBB8)
  A348   THEN SET UP BOTTOM AS HIMEM MSB (BBB9)
  A34B   GET OLD ORIGINAL HIMEM (BEFORE ANY BUFFERS) (BEFB)
```

```
BASIC Interpreter (BI) -- V1.0.1 -- 1 JAN 84    NEXT OBJECT ADDR: A34E
-------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------

  A34E   SAME AS THIS ONE?
  A350   THEN NOTHING ELSE TO DO >>A38B
  A352   ASSUME NO BUFFERS BY REPLACING OLD HIMEM
  A354   ANY EXEC FILE OPEN? (BE43)
  A357   NO, CONTINUE >>A35E
  A359   YES, MOVE EXEC BUFFER TO OLD HIMEM <A32D>
  A35C   AND GO MOVE HIMEM DOWN BY $400 >>A37C
  A35E   ELSE, START WITH TOP BUFFER  (BBB8)
  A361   ANY OPEN FILES? (BE4D)
  A364   IF NOT, WE ARE DONE >>A388
  A366   SEARCH FOR OPEN FILE WITH THIS BUFFER (BC93)
  A369   NOT IT? >>A385
  A36B   GOT IT, GIVE IT NEW HOME AT HIMEM
  A36D   AND SET BUFFER LOW <A38D>
  A370   THEN TO NEW LOC <A2D6>
  A374   DROP TOP BUFFER PTR BY $400 (BBB8)
  A37C   AND DROP HIMEM BY $400
  A383   AND GO DO NEXT BUFFER >>A35E
  A385   ---
  A386   (LOOP TO SEARCH FOR OPEN FILES) >>A366
  A388   WHEN FINISHED, GARBAGE COLLECT <A07B>
  A38B   ---
  A38C    THEN EXIT NORMALLY TO CALLER

          ******** SET BUFFER BELOW ALL OTHERS ***

  A38D   ---
  A38E   USE BOTTOM BUFFER PTR (BBB9)
  A391   SET FILE BUFFER <A2D6>
  A395    AND EXIT

A396 *********** COPY BLOCK DOWN IN MEMORY ****************************

  A396   COPY ALL FULL PAGES DOWN TO THEIR NEW HOME
  A39D   COPYING $3A-->$3C
  A3A4   BUMP BOTH MSB'S
  A3A8   DROP PAGE COUNTER (BC93)
  A3AB   AND CONTINUE >>A39D
  A3AD   NO SHORT LAST PAGE? (BC92)
  A3B0   THEN EXIT NOW >>A3B9
  A3B2   ELSE, COPY PARTIAL PAGE
  A3B9    THEN EXIT

A3BA *********** COPY BLOCK UP IN MEMORY ******************************

  A3BA   PARTIAL PAGE? (BC92)
  A3BD   NO, JUST COPY FULL PAGES NOW >>A3C6
  A3BF   YES, COPY SHORT PAGE FIRST <A3D1>
  A3C2   DROP BOTH MSB'S
  A3C6   PAGE COUNT GONE TO ZERO? (BC93)
  A3C9   YES, DONE  >>A3D9
```

```
BASIC Interpreter (BI) -- V1.0.1 -- 1 JAN 84    NEXT OBJECT ADDR: A3CB
-------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------

  A3CB  ELSE, DROP PAGE COUNT (BC93)
  A3CE  AND GO COPY A FULL PAGE UP >>A3BF

  A3D1  ---
  A3D2  COPY REMAINDER OF PAGE UP (BACKWARDS)
  A3D9  RETURN

A3DA ********** ADJUST ALL STRING ADDRS ****************************
         (BC87 HAS ADDITIVE ADJUSTMENT FACTOR)

  A3DA  USE LOMEM PAGE AS MSB FOR $3E/3F
  A3DE  GET LOMEM LSB
  A3E0  AND END OF SIMPLE VARS PAGE
  A3E3  JUMP INTO THE LOOP >>A3EA
  A3E5  ---
  A3E6  SKIP ONE SIMPLE VARIABLE
  A3EA  ---
  A3EC  OVERFLOW? >>A3F0
  A3EE  YES, BUMP MSB
  A3F0  FINISHED WITH SIMPLE VARS?
  A3F4  (CHECK BOTH MSB AND LSB OF PTR)
  A3F6  ---
  A3F7  YES... >>A40D
  A3F9  NO,
  A3FB  LOOK AT A SIMPLE VARIABLE
  A400  SKIP INTEGER AND REAL VARS >>A3E5
  A402  (DOUBLE CHECK MSB)
  A406   ITS A STRING, POINT TO ITS LEN/ADDR
  A407  ADJUST IT IF NECESSARY <A435>
  A40A  THEN SKIP OVER IT >>A3E5

  A40D  COPY ARRAYS STARTING LSB
  A40F  (MSB IS IN X REGISTER NOW) (BC81)
  A412  ---
  A413  FIND A STRING ARRAY <A199>
  A416  NO MORE? THEN DONE... >>A434
  A418  ---
  A41B  ADJUST ITS ADDRESS IF NEED BE <A435>
  A421  SKIP TO NEXT STRING ELEMENT OF ARRAY
  A429  AT END OF THIS ARRAY YET? (BC81)
  A42C  NO... >>A418
  A42E  (CHECK MSB ALSO)
  A432  YES..., GO GET NEXT ARRAY  >>A412
  A434  RETURN

A435 ********** ADJUST A STRING ADDRESS ****************************
```

```
BASIC Interpreter (BI) -- V1.0.1 -- 1 JAN 84    NEXT OBJECT ADDR: A435
-------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------

  A435  GET STRING LENGTH
  A437  IGNORE NULL STRINGS >>A448
  A439   POINT TO MSB OF ADDRESS
  A43B  IS STRING STORED OUTSIDE OF PROGRAM?
  A43F  NO, LEAVE IT ALONE >>A448
  A441  STORE ABOVE LOMEM, ADD FACTOR TO MSB
  A448   THEN EXIT

A449 ********** COMPRESS ALL ASOFT VARS ****************************
         THIS ROUTINE SQUASHES ALL APPLESOFT VARS
         UP AGAINST THE BOTTOM OF THE STRINGS
         HIMEM --> -------------------------
                   |                       |
                   |       STRINGS         |
                   |                       |
                   -------------------------
                   |                       |
                   |       ARRAY VARS      |
                   |                       |
                   -------------------------
                   |                       |
                   |       SIMPLE VARS     |
                   |                       |
                   -------------------------

  A449  GARBAGE COLLECT FIRST <A054>
  A44C  ERROR? >>A4AE
  A44E  COMPUTE LENGTH OF SIMPLE AND ARRAY VARS
  A453  AND SAVE IT (BC89)
  A463  NEXT, COMPUTE LENGTH OF SIMPLE VARS ONLY
  A467  AND SAVE IT (BC8B)
  A471  SUBTRACT VAR LENGTH FROM STRING START
  A473  TO FIND A PLACE TO PUT THE VARS UNDER (BC92)
  A476  THE STRINGS (START ON AN EVEN PAGE BOUND)
  A47C  $3C/$3D --> PLACE TO PUT VARS
  A483  $3A/$3B --> START OF VARS (ROUNDED TO EVEN
  A485                PAGE ALIGNEMENT)
  A48B  COPY VARS UP AGAINST STRINGS <A3BA>
  A490  STORE START OF VARS PTR (BC8E)
  A496  BUMPING PAGE NUMBER BY ONE
  A4A0  SUBTRACT THIS PTR FROM HIMEM TO COMPUTE (BC90)
  A4A3  TOTAL LENGTH OF COMBINED VARS/STRINGS
  A4A5  AND SAVE THIS TOO (BC8D)
  A4A8  ALSO, SAVE HIMEM MSB IN CASE THEY ARE MOVED
  A4AE   DONE, EXIT
```

```
BASIC Interpreter (BI) -- V1.0.1 -- 1 JAN 84    NEXT OBJECT ADDR: A4AE
------------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
------------------------------------------------------------------------

A4AF ********** REEXPAND COMPRESSED VARS ***************************
         THIS ROUTINE MOVES SIMPLE AND ARRAY VARS
         BACK DOWN TO LOMEM.
         HIMEM --> -------------------------
                   |                       |
                   |       STRINGS         |
                   |                       |
                   -------------------------
                   |                       |
                   |                       |
                   /     FREE SPACE        /
                   /                       /
                   |                       |
                   |                       |
                   -------------------------
                   |                       |
                   |       ARRAY VARS      |
                   |                       |
                   -------------------------
                   |                       |
                   |       SIMPLE VARS     |
                   |                       |
         LOMEM --> -------------------------

  A4AF  SAVE LENGTH OF SIMPLE AND ARRAY VARS (BC89)
  A4B6  ADD THIS TO START OF COMPRESSED VARS PTR
  A4B8  TO GIVE START OF STRINGS ($6D/$6E)
  A4C4  $3C/$3D --> LOMEM (WHERE TO PUT SIMPLE VARS)
  A4CB  $6B/$6C --> WHERE TO PUT ARRAY VARS
  A4D6  $3A/$3B --> START OF COMPRESSED VARS (BC8E)
  A4E0  COPY SIMPLE/ARRAY VARS DOWN TO LOMEM <A396>
  A4E7  COMPUTE START OF STRINGS BY ADDING VARS
  A4E9  LENGTH TO VARS ORIGIN
  A4F2  DID HIMEM MOVE SINCE VARS WERE COMPRESSED?
  A4F7  NO... >>A4FF
  A4F9  YES, ADJUST BY DIFFERENCE IN HIMEM'S (BC87)
  A4FC  GO ADJUST ALL STRING POINTERS <A3DA>
  A4FF   THEN EXIT
  A500  RETURN

A501 ********** FORMAT CATALOG ENTRY LINE ***************************

  A501  PUT OUT A BLANK LINE <A6A9>
  A504  DOUBLE QUOTE TO $200
  A509  GET LENGTH OF NAME (0259)
  A50F  COPY NAME TO LINE (0259)
  A51A  ZERO ACCUMULATOR FOR LATER (BCB1)
  A51D  GET FILE TYPE (0269)
  A520  I KNOW OF ONLY 13
  A522  ---
```

```
BASIC Interpreter (BI) -- V1.0.1 -- 1 JAN 84    NEXT OBJECT ADDR: A524
------------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
------------------------------------------------------------------------

  A524  LOOK UP FILE TYPE IN TABLE (B9DB)
  A527  FOUND IT? >>A534
  A52D  FILE TYPE NOT IN MY TABLE
  A52F  PRINT IT IN HEXADECIMAL <A64F>
  A532  AND CONTINUE BELOW >>A578
  A534  ELSE, FOR KNOWN TYPES
  A537  COPY NAME OF TYPE TO THE LINE (B9E9)
  A542  80 COLUMNS PER LINE? (BCB6)
  A545  YES... >>A590
  A547  NO,
  A549  BIN FILE?
  A54B  YES... >>A562
  A54D  TXT FILE?
  A54F  NO... >>A578
  A551  YES, R VALUE GIVEN AS SUBTYPE
  A55C  CONVERT R VALUE TO DECIMAL <A66C>
  A55F  SKIP OVER BIN CODE >>A573
  A562  BIN FILE, USE AD VALUE AS SUBTYPE
  A56A  CONVERT IT TO TWO HEX DIGITS <A64F>
  A573  ADD AN "=" SIGN
  A578  COPY MSB OF END OF FILE MARK (0270)
  A586  CONVERT LOW TWO BYTES OF EOF <A66C>
  A58D  DO CREATION DATE/TIME <A5AD>
  A590  ---
  A598  CONVERT BLOCKS USED <A66C>
  A5A0  CHECK FOR WRITE ACCESS
  A5A2  UNLOCKED? >>A5A9
  A5A4  NO, ADD A "*"
  A5A9  FALL THRU TO DO LAST MODIFIED DATE/TIME
  A5AB  AND THEN EXIT TO CALLER

A5AD ********** FORMAT A DATE/TIME ***********************************
          X = OFFSET FROM $259 TO FIELD
          Y = $201 OFFSET TO DATE/TIME VALUE

  A5AD  ISOLATE YEAR (025A)
  A5B1  AND STORE IT (BCB5)
  A5B8  ISOLATE DAY
  A5BA  AND STORE IT (BCB4)
  A5BE  ISOLATE MONTH
  A5C4  (MONTH = 0 IS NO GOOD) >>A5E0
  A5C8  (MONTH > 12 IS ALSO BAD) >>A5E0
  A5CA  STORE MONTH (BCB3)
  A5CE  MULTIPLY MONTH INDEX BY 3 (BCB3)
  A5D1  AND SAVE IT INSTEAD (BCB3)
  A5D7  (DAY = 0 IS NO GOOD) >>A5E0
  A5DE  (YEAR MUST BE < 99) >>A5F2
```

```
BASIC Interpreter (BI) -- V1.0.1 -- 1 JAN 84    NEXT OBJECT ADDR: A5DE
-------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------

  A5E0   OTHERWISE, BAD DATE!
  A5E2  BACK UP 6 CHARACTERS ON LINE
  A5E7  AND PRINT "<NO DATE>" (BA37)
  A5F1   THEN EXIT RIGHT AWAY

  A5F2  DATE OK, GET HOUR (0025C)
  A5F6  ANO MINUTES (0025B)
  A5FB  MINUTES > 60?
  A5FD  NO.. >>A600
  A5FF   YES, USE ZERO MINUTES
  A600  CONVERT MINUTES (LEFT ZERO FILL) <A647>
  A605  THEN PRINT A ":" (0201)
  A609   GET HOUR AGAIN
  A60C  GREATER THAN 24 HOURS?
  A60E  NOPE >>A611
  A610   YES, USE ZERO
  A611  10 OR MORE HOURS (TWO DIGITS?)
  A614  IN ANY CASE, CONVERT HOURS <A66C>
  A618  IF TWO DIGITS... >>A61B
  A61A   IF ONE, AOJUST LINE PTR
  A61B  ---
  A61F  CONVERT YEAR (LEFT ZERO FILL) <A647>
  A623  GET MONTH INDEX (*3) (BCB3)
  A626   POINT TO LAST CHARACTER
  A629  COPY MONTH NAME FROM TABLE (BA0F)
  A62C  TO LINE (0201)
  A634  BACKWARDS... >>A629
  A638  PUT A "-" IN (0201)
  A63B  TWO PLACES (0205)
  A644  EXIT BY CONVERTING DAY >>A66C

A647 *********** CONVERT 2 DIGIT NUMBER ******************************
         (FORCE LEFT ZERO FILL)

  A647  ---
  A648  ADD 100 TO FORCE SIGNIFICANCE IN TENS
  A64A  CONVERT IT <A66C>
  A64D   IGNORE 100'S PLACE
  A64E  RETURN

A64F *********** CONVERT TO HEX ***************************************

  A64F  ---
  A650  ISOLATE LOW NIBBLE
  A652  AND GO CONVERT IT FIRST <A65A>
  A656  NOW ISOLATE HIGH NIBBLE
  A659  ANO FALL THRU TO CONVERT IT ALSO
```

```
BASIC Interpreter (BI) -- V1.0.1 -- 1 JAN 84    NEXT OBJECT ADDR: A659
-------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------

  A65A  CONVERT NIBBLE TO NUMERIC ASCII
  A65C  >9?
  A65E  NO >>A662
  A660  YES, CONVERT $BA-$BF TO $C1-$C6
  A662  AND STORE THE RESULT (0201)
  A665   BUMP LINE INOEX BACK
  A666  PRECEED WITH A $ SIGN
  A66B  RETURN

A66C *********** CONVERT TO DECIMAL ***********************************

  A66C  A,X = NUMBER    Y=INDEX TO LAST FIELO DIGIT (BCB0)
  A66F  STORE NUMBER IN ACCUMULATOR (BCAF)
  A672  DIVIDE BY 10 <A68A>
  A675  GET DIGIT ANO CONVERT IT (BCB2)
  A67A  STORE IN LINE (0201)
  A67D   AND OROP LINE INDEX BY ONE
  A67E  IS QUOTIENT NOW ZERO? (BCAF)
  A687  NO, CONTINUE UNTIL IT IS >>A672
  A689   ELSE, EXIT

         ******** DIVIOE ACCUMULATOR BY 10 *******

  A68A  24 BIT SHIFT (3 BYTES)
  A68E  CLEAR SUM (BCB2)
  A691  GO ROL ACCUMULATOR LEFT ONE BIT  <AB17>
  A694  ALSO ROL 4TH BYTE OF ACCUM (BCB2)
  A698  IF MSB > 10... (BCB2)
  A6A2  THEN ADD ONE TO ACCUMULATIVE SUM (BCAF)
  A6A5  ---
  A6A6  SHIFT 24 TIMES >>A691
  A6A8  RETURN
  A6A9  ---
  A6B3  RETURN

A6B4 *********** SYNTAX: PARSE COMMAND LINE ***************************
          (ALSO EXTERNAL ENTRY FOR COMMAND STRINGS)

  A6B4  INIT COMMAND NUMBER TO -1
  A6BB  A BLANK ENDS EACH STRING (BCA9)
  A6C0  AT MOST 8 CHARACTERS IN A COMMAND (BCAA)
  A6C3  PARSE COMMAND ITSELF <AA5B>
  A6C6  GET FIRST LETTER (BCBD)
  A6C9  MUST BE ALPHABETIC
  A6CB  IT IS... >>A6D4
  A6CD  IT'S NOT, IS IT A "-"?
  A6CF  YES, OK THEN... >>A6D4
  A6D1  ELSE, ITS BAD - SYNTAX ERROR >>A879
  A6O4  SCAN FOR COMMAND IN TABLES <AB21>
  A6D7  BAD COMMAND? >>A6D1
```

```
BASIC Interpreter (BI) -- V1.0.1 -- 1 JAN 84    NEXT OBJECT ADDR: A6D9
------------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
------------------------------------------------------------------------

  A6D9  NO, IMMEDIATE COMMAND MODE? (BE42)
  A6DC  NO, DEFERRED... >>A6E9
  A6DE  IMMEDIATE, EXEC ACTIVE? (BE43)
  A6E1  YES, NEVER MIND >>A6E9
  A6E3  ERASE TO END OF LINE <FC9C>
  A6E6  AND GO TO A NEW LINE ON SCREEN <9FE2>
  A6E9  ASSUME NO PARMS AT ALL
  A6F1  NO PATH NAME YET (BCBD)
  A6F4  NO SECONDARY PATH NAME EITHER (0280)
  A6FA  CURRENT SLOT = DEFAULT SLOT (BE61)
  A700  CURRENT DRIVE = DEFAULT DRIVE (BE62)
  A705  BUFFER ALLOCATION = HIMEM (BC88)
  A708  GET LENGTH OF COMMAND NAME (BE52)
  A70D  ALLOW 2 MORE CHARACTERS FOR NOW (BCAA)
  A710  ARE ANY PARAMETERS PERMITTED? (BE54)
  A713  NO...MUST BE MON OR NOMON >>A776
  A715  YES, IN# OR PR#?
  A716  YES...  >>A779
  A718  ELSE, REPARSE THE COMMAND <AA5B>
  A71D  FOR THIS COMMAND... (BE54)
  A720  DOES THE PREFIX NEED FETCHING? >>A727
  A722  YES,
  A724  MLI: GET PREFIX FROM DEFAULT DRIVE <BE70>
  A727  ---
  A729  END OF LINE? >>A776
  A72B  NO, COMMA?
  A72D  NO >>A732
  A72F  YES, NO FILENAME, LOOK FOR KEYWORDS >>A7C7
  A732  "/"?
  A734  YES >>A73A
  A736  NO, ALPHABETIC?
  A738  NO...FILE NAMES MUST BEGIN THAT WAY >>A76F
  A73A  ---
  A73B  DON'T FLUSH ANY BLANKS OUT OF PATHNAME
  A740  ALLOW 64 CHARACTERS NEXT PARSE
  A746  PARSE NEXT OPERAND ON LINE <AA5F>
  A74A  SAVE ITS LENGTH (BCBC)
  A74F  FOUND A PATHNAME#1 (BE56)
  A755  COPY PARM KEYWORD TO $280 (BCBC)
  A758  (ASSUMING PATHNAME1=PATHNAME2) (0280)
  A75F  CHECK NEXT CHAR (OTHER THAN A BLANK) <AA7A>
  A762  NOT COMMA OR RETURN, BAD! >>A76C
  A764  RETURN? >>A7D8
  A766  NO, PATHNAME EXPECTED NOW? (BE54)
  A76A  YES, ALL IS WELL >>A7A2
  A76C  NO, "SYNTAX ERROR" >>A879
  A76F  NON ALPHA FILE NAME, CHECK COMMAND NUMBER (BE53)
  A772  IS IT "RUN"
  A774  NO, ERROR >>A76C
  A776  YES, ITS OK THEN (MIGHT BE "RUN 100") >>A7D8
  A779  IN#S/PR#S, REPARSE COMMAND <AA5B>
```

```
BASIC Interpreter (BI) -- V1.0.1 -- 1 JAN 84    NEXT OBJECT ADDR: A77C
------------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
------------------------------------------------------------------------

  A77C  RETURN FOUND - ERROR >>A76C
  A77E  "A"?  (ADDRESS KEYWORD)
  A780  IF SO, GO PARSE THAT KEYWORD ONLY >>A7CC
  A782  ELSE, ZERO ACCUMULATOR <AB77>
  A785  CONVERTING ONE BYTE'S WORTH (BCAD)
  A78A  PUT IT IN PR#/IN# SLOT VALUE AREA (BCAE)
  A78F  FOUND SLOT FOR PR#/IN# (BE56)
  A792  CONVERT SLOT # <A9A0>
  A795  ERROR? >>A7A1
  A797  GET CONVERTED VALUE (BE6B)
  A79A  >8?
  A79C  NO, ITS OK  >>A7D1
  A79E  YES, "RANGE ERROR"
  A7A1  RETURN
  A7A2  SECOND PATHNAME EXPECTED?
  A7A3  NO >>A7C7
  A7A5  YES, FLUSH TO NON-BLANK <AA7A>
  A7A8  NOTHING ELSE ON LINE???? >>A76C
  A7AB  DON'T FLUSH ANY BLANKS OUT OF PATHNAME
  A7B2  COPY SECOND PATHNAME TO $281 <AA40>
  A7B7  SAVE IT'S LENGTH (LESS 1) (0280)
  A7BC  FOUND PATHNAME1 AND PATHNAME2 (BE56)
  A7C0  GET LAST CHARACTER AGAIN <AA7A>
  A7C3  IF NOT COMMA OR RETURN, "SYNTAX ERROR" >>A76C
  A7C5  RETURN? >>A7D8
  A7C7  NO, COMMA, FLUSH TO NON-BLANK <AA7A>
  A7CA  SYNTAX ERROR IF TWO COMMAS IN A ROW >>A76C
  A7CC  LOOKUP KEYWORD CHAR AND PARSE ITS VALUE <A928>
  A7CF  EXIT NOW? >>A7A1
  A7D1  NO, FLUSH TO NON-BLANK <AA7A>
  A7D4  SYNTAX ERROR IF COMMA OR RETURN NOT FOUND >>A76C
  A7D6  COMMA? YES, GO GET NEXT KEYWORD >>A7C7

  A7D8  GET PARSED SLOT (BE61)
  A7DB  MUST BE NON-ZERO >>A79E
  A7DD  AND LESS THAN 8
  A7DF  OR ELSE - "RANGE ERROR" >>A79E
  A7E1  CHECK DRIVE TOO (BE62)
  A7E6  MUST BE EITHER 1 OR 2
  A7ED  IS THIS A DEFERRED COMMAND?
  A7F0  NO... >>A7FB
  A7F2  YES, IS A PROGRAM RUNNING? (BE42)
  A7F5  YES >>A7FB
  A7F7  NO, "NOT DIRECT COMMAND"
  A7FA  RETURN
  A7FB  EXPECTING NO PATHNAMES? >>A83D
  A7FD  NO... (BE55)
  A800  ARE S AND D VALID FOR THIS CMD?
  A802  NO >>A83D
  A804  YES, HAVE WE GOT PATHNAME1? (BE56)
  A808  YES >>A813
```

```
BASIC Interpreter (BI) -- V1.0.1 -- 1 JAN 84    NEXT OBJECT ADDR: A80D
------------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
------------------------------------------------------------------------

  A80D  IS PATHNAME REQUIRED?
  A80F  YES, "SYNTAX ERROR" >>A879
  A811  NO, OPTIONAL - NO PREFIX FETCH THEN >>A83D
  A816  DOES PATHNAME1 START WITH A "/"?
  A818  YES, FULLY QUALIFIED  >>A81F
  A81A  NO, IS THERE A PREFIX ACTIVE? (BF9A)
  A81D  NO >>A838
  A81F  YES, (BE57)
  A822  SLOT/DRIVE GIVEN WITH THIS COMMAND?
  A824  NO, FORGET IT >>A83D
  A826  YES, DO WE HAVE PATHNAME ALSO? >>A838
  A828  NO,
  A82A  NULL OUT PATHNAME1 (BCBC)
  A832  MARK THAT WE WILL HAVE ONE SOON (BE56)
  A838  ADD PREFIX TO FILENAMES <A87D>
  A83B  ERROR? >>A87B
  A83D  GET COMMAND NUMBER (BE53)
  A840  *2 AS INDEX INTO TABLE
  A842  GET ADDRESS OF COMMAND HANDLING ROUTINE (B93F)
  A84B  AND STORE IT FOR INDIRECT JMP (BCAC)
  A850  EXTERNAL COMMAND? IF SO GO NOW! >>A876
  A852  MY OWN COMMAND, "PREFIX"?
  A854  YES, GO NOW >>A876
  A859  S OR D VALID KEYWORDS FOR THIS CMD?
  A85B  NO, GO NOW >>A876
  A860  PATHNAME1 GIVEN WITH THIS COMMAND?
  A861  NO, GO NOW >>A876
  A863  YES, GET FILE INFO FOR PATHNAME1 <B82A>
  A866  NO ERRORS I HOPE >>A876
  A868  ERROR WAS PATH NOT FOUND?
  A86A  NO, REAL ERROR - SAY SO >>A87B
  A86F  CAN WE CREATE PATHNAME1?
  A871  YES, OK THEN >>A876
  A873  ELSE, "PATH NOT FOUND"
  A875  RETURN
  A876  GO TO COMMAND HANDLING ROUTINE >>BCAB

A879 *********** SYNTAX ERROR ***************************************

  A879  LOAD BI CODE FOR "SYNTAX ERROR"
  A87B   AND RETURN WITH ERROR CONDITION
  A87C  RETURN

A87D *********** ADD PREFIX TO PATHNAMES ****************************

  A87D  GET SLOT NUMBER (BE61)
  A884  PUT SLOT IN HIGH 3 BITS
  A886  ADD DRIVE TO TOP BIT AND SHIFT SLOT DOWN (BE62)
  A88E  ...TO FORM THE UNIT NUMBER (BEC7)
  A893  READ THE PATHNAME PREFIX TO $201 (BEC8)
  A89D  MLI: ONLINE <BE70>
```

```
BASIC Interpreter (BI) -- V1.0.1 -- 1 JAN 84    NEXT OBJECT ADDR: A8A0
------------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
------------------------------------------------------------------------

  A8A0  ERROR? >>A87B
  A8A5  DEFAULT DRIVE = PARSED DRIVE (BE3D)
  A8AB  DEFAULT SLOT = PARSED SLOT (BE3C)
  A8B1  PATHNAME1 STARTS WITH "/"?
  A8B3  THEN ITS ALREADY GOT A PREFIX >>A926
  A8B8  ELSE, GET LENGTH OF PATHNAME
  A8BA  BUMP IT BY 2 (TO ALLOW FOR /'S)
  A8C2  WITH PREFIX WILL IT EXCEED 64 CHARS?
  A8C7  YES, "SYNTAX ERROR" >>A927
  A8C9  NO, UPDATE LENGTH TO INCLUDE PREFIX (BCBC)
  A8CF  ---
  A8D3  AND COPY PATHNAME1 FORWARD TO MAKE ROOM (BCBD)
  A8DC  PUT A "/" AT THE BEGINNING
  A8E1  AND AT THE END (BCBD)
  A8E4  COPY PREFIX JUST READ TO START OF PATHNAME1 (0200)
  A8EA  GET COMMAND NUMBER (BE53)
  A8ED  "OPEN"?
  A8EF  YES, DONE NOW! >>A926
  A8F1  "APPEND"?
  A8F3  YES, DONE NOW! >>A926
  A8F5  "EXEC"?
  A8F7  YES, DONE NOW! >>A926
  A8F9  ELSE, GET LENGTH OF PATHNAME2 (0280)
  A8FE  COMBINE THIS WITH PREFIX LENGTH (0201)
  A901  MORE THAN 64 CHARS?
  A906  IF SO, "SYNTAX ERROR" >>A927
  A908  UPDATE LENGTH (0280)
  A90B  ---
  A90F  COPY PATHNAME2 FORWARD TO MAKE ROOM (0281)
  A918  PUT A "/" IN FIRST
  A91D  THEN THE PREFIX AND ANOTHER SLASH (0281)
  A926  ---
  A927   DONE!

A928 *********** KEYWORD LOOKUP ***************************************

  A928  ZERO THE ACCUMULATOR <AB77>
  A92B  NINE POSSIBLE KEYWORDS IN TABLE
  A92D  COMPARE AGAINST EACH (B9BD)
  A930  FOUND IT? >>A967
  A935  NO, IS IT "T"? (FILE TYPE)
  A937  YES, OK THEN >>A93C
  A939  ELSE, BAD KEYWORD >>A879
  A93C  IT'S "T", IS IT PERMITTED ON THIS CMD?
  A941  NO, ERROR >>A963
  A946  ELSE, MARK WE HAVE "T" (BE56)
  A94B  START WITH TYPE INDEX OF 0 (BCAD)
  A950  INDICATE WHERE T VALUE IS TO GO (BCAE)
  A953  AND GO PARSE ONE CHAR <AA7A>
  A956  NOTHING THERE??? >>A939
  A958  IS IT A $?
```

```
BASIC Interpreter (BI) -- V1.0.1 -- 1 JAN 84    NEXT DBJECT ADDR: A95A
-------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------

  A95A  YES, HE GAVE TYPE IN HEX >>A9B6
  A95C  IS IT ALPHABETIC?
  A95E  NO, CONVERT DECIMAL TYPE >>A9A0
  A960  ELSE, GO LOOKUP TYPE NAME IN TABLE >>A9F6

  A963  ---
  A964  "INVALID PARAMETER"
  A966  RETURN

  A967  GET BIT POSITION OF THIS KEYWORD (B9C7)
  A96A  IGNORE "V" >>A987
  A96C  IS THIS KEYWORD PERMITTED? (BE55)
  A96F  NO, NOT WITH THIS COMMAND ANYWAY >>A963
  A971  S OR D?
  A973  NO >>A981
  A975  YES, ALREADY FDUND IT DN THIS LINE? (BE57)
  A978  YES, DON'T CHANGE DRIVE DEFAULT >>A987
  A97A  ELSE, ASSUME DRIVE = 1
  A981  MARK WE HAVE SLOT/DRIVE (BE57)
  A987  GET SIZE-1 IN BYTES OF VALUE  (B9D1)
  A994  AND OFFSET TO VALUE IN STORAGE AREA (BCAE)
  A997  FLUSH TO NON-BLANK <AA7A>
  A99A  NOTHING ELSE THERE? >>A9F0
  A99C  IS NEXT CHAR A "$"?
  A99E  YES, GO CONVERT HEX - ELSE, FALL THRU >>A9B6

A9A0 *********** CONVERT DECIMAL NUMBER ******************************

  A9A0  SAVE LINE INDEX (BE4B)
  A9A3  CONVERT/ADD ONE DECIMAL DIGIT TO ACCUM <AA9C>
  A9A6  OK.. >>A9AC
  A9A8  OVERFLOW? THEN "RANGE ERROR" >>A9F3
  A9AA  BAD DIGIT? THEN "SYNTAX ERROR" >>A9F0
  A9AC  RESTORE LINE INDEX (BE4B)
  A9AF  FLUSH TO NEXT NON-BLANK <AA7A>
  A9B2  AND GO BACK TO CONVERT NEXT DIGIT >>A9A0
  A9B4  ALL DONE, END OF LINE OR COMMA >>A9CF

A9B6 *********** CONVERT HEX NUMBER **********************************

  A9B6  FLUSH TO NEXT NON-BLANK (SKIP "$") <AA7A>
  A9B9  NOTHING LEFT? >>A9F0
  A9BB  SAVE LINE INDEX (BE4B)
  A9BE  CONVERT HEX DIGIT <AAEE>
  A9C1  OK.. >>A9C7
  A9C3  OVERFLOW? THEN "RANGE ERROR" >>A9F3
  A9C5  BAD DIGIT? THEN "SYNTAX ERROR" >>A9F0
  A9C7  RESTORE LINE INDEX (BE4B)
  A9CA  FLUSH TO NEXT NON-BLANK <AA7A>
  A9CD  AND GO CONVERT NEXT DIGIT >>A9BB
```

```
BASIC Interpreter (BI) -- V1.0.1 -- 1 JAN 84    NEXT OBJECT ADDR: A9CD
-------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------

A9CF *********** STORE KEYWORD VALUE *********************************

  A9CF  HOW MANY BYTES TO CHECK?
  A9D4  ALL HAVE BEEN CHECKED? >>A9DE
  A9D6  ND, INSURE MSB'S DF ACCUM ARE ZERO (BCAF)
  A9D9  IF NUMBER IS A SHORT INTEGER >>A9F3
  A9E1  COPY ACCUM TO PROPER PARM STDRAGE CELL (BCAF)
  A9EB  RESTORE LINE INDEX (BE4B)
  A9EF   AND EXIT

  A9F0  "SYNTAX ERRDR" JUMP >>A879
  A9F3  "RANGE ERROR" JUMP >>A79E

A9F6 *********** STORE KEYWORD VALUE *********************************

  A9F6  ---
  A9F8  COPY 3 CHARACTER TYPE TO ACCUM (BCAF)
  A9FE  (COPIED ALL 3?) >>AA07
  AA00  (GET NEXT CHAR IGNORING BLANKS) <AA7A>
  AA05  MUST HAVE 3 CHARACTERS! >>A9F0
  AA07  SAVE LINE INDEX (BE4B)
  AA0A  INITIALIZE NAME INDEX TO ZERO
  AA0F  HAVE ALL 13 BEEN CHECKED?
  AA11  YES, NO MATCH >>A9F0
  AA14  ELSE, INDEX*3 (BCAD)
  AA18  COMPARE TYPE GIVEN (BCAF)
  AA1B  TO TYPES IN TABLE (B9E9)
  AA1E  (IGNORE MSB'S)
  AA1F  NO MATCH ALREADY... >>AA29
  AA23  ELSE,
  AA25  CHECK ALL THREE CHARS >>AA18
  AA27  THEY ALL MATCH! WE FOUND IT >>AA2E
  AA29  NOT THE RIGHT ONE, (BCAD)
  AA2C  GO TRY THE NEXT ONE >>AA0A
  AA2E  REVERSE NAME INDEX
  AA35  AND GET TYPE VALUE FROM TABLE (B9DB)
  AA38  STORE IT IN TYPE VALUE STORAGE AREA (BE6A)
  AA3B  RESTORE LINE INDEX (BE4B)
  AA3F   AND EXIT

AA40 *********** COPY PATHNAME2 ***********************************

  AA40  GET NEXT CHARACTER <AA8A>
  AA43  AND STORE IT INDEXED OFF $280 (0280)
  AA47  COMMA?
  AA49  YES, DONE >>AA77
  AA4B  BLANK?
  AA4D  YES, DONE >>AA77
  AA4F  RETURN?
  AA51  YES, OUT NOW >>AA88
```

```
BASIC Interpreter (BI) -- V1.0.1 -- 1 JAN 84    NEXT OBJECT AODR: AA53
------------------------------------------------------------------------
AOOR    OESCRIPTION/CONTENTS
------------------------------------------------------------------------

  AA53  PATHNAME TOO LONG? (BCAA)
  AA56, NO, CONTINUE COPYING >>AA40
  AA58  ELSE, SET NOT-EQUAL CONOITION
  AA5A   ANO EXIT

AA5B *********** COPY COMMANO NAME INTO TXTBUF ***********************

  AA5B  SET INOICIES
  AA5F  GET NEXT NON-BLANK <AA8A>
  AA62  COPY TO TXTBUF (BCBD)
  AA66  COMMA?
  AA68  YES, DONE >>AA77
  AA6A  BLANK?
  AA6C  YES, DONE >>AA77
  AA6E  RETURN?
  AA70  YES, OONE >>AA88
  AA72  AT MAX LENGTH (8)? (BCAA)
  AA75  NO, CONTINUE >>AA5F
  AA77  ELSE, SET NOT-EQUAL CONOITION
  AA79   ANO EXIT

AA7A *********** FLUSH TO NON-BLANK **********************************
          Z-FLAG SET IF COMMA OR RETURN FOUNO
          C-FLAG SET IF COMMA

  AA7A  IGNORE BLANKS
  AA7F  GET NEXT NON-BLANK <AA8A>
  AA82  COMMA?
  AA84  YES, OUT >>AA89
  AA86  RETURN?
  AA88   EXIT INOICATING WHAT WE FOUNO
  AA89  RETURN

AA8A *********** GET NEXT CHARACTER **********************************

  AA8A  GET NEXT CHAR IN INPUT LINE (0200)
  AA8O  FORCE OFF MSB
  AA8F  LOWER CASE?
  AA91  NO >>AA95
  AA93  YES, FORCE UPPER CASE
  AA95   BUMP LINE INOEX
  AA96  IS THIS A FLUSH CHARACTER (LIKE BLANK)? (BCA9)
  AA99  YES, GO GET NEXT ONE >>AA8A
  AA9B   ELSE, RETURN WITH IT

AA9C *********** CONVERT OIGIT AND ADD TO ACCUM **********************
```

```
BASIC Interpreter (BI) -- V1.0.1 -- 1 JAN 84    NEXT OBJECT AODR: AA9C
------------------------------------------------------------------------
AOOR    DESCRIPTION/CONTENTS
------------------------------------------------------------------------

  AA9C  NUMERIC?
  AA9E  NO >>AAA4
  AAA2  YES >>AAA8
  AAA4   NOT NUMERIC, EXIT WITH CARRY SET
  AAA5  ANO Z-FLAG RESET
  AAA7  RETURN
  AAA8  ISOLATE DECIMAL PORTION OF DIGIT
  AAAB  CURRENT VALUE OF ACCUM... (BCB1)
  AAAE  >1,703,936?
  AABO  YES, OVERFLOW >>AAO4
  AAB4  PUSH ENTIRE ACCUM ONTO STACK (BCAF)
  AABB  ACCUM*2 (ROL IT ONCE) <AB17>
  AABE  ACCUM*4 (ANO AGAIN) <AB17>
  AAC4  ---
  AAC5  ACCUM*4+ACCUM --> ACCUM*5 (BCAF)
  AAO1  FINALLY, ACCUM*5*2 --> ACCUM*10 <AB17>
  AAD4  ---
  AAO5  ACCUM OVERFLOW? >>AAEA
  AAO7  NO, ADO NEW OIGIT TO ACCUM (BCAF)
  AAOA  ANO STORE IT (BCAF)
  AAOO  NO CARRY? >>AAEO
  AAE0  GOT CARRY, PROPAGATE IT THRU ACCUM (BCB0)
  AAEA  OVERFLOW ERROR
  AAEO   NORMAL EXIT

AAEE *********** CONVERT HEX OIGIT ANO AOO ***************************

  AAEE  NUMERIC?
  AAF0  NO >>AAFE
  AAF4  YES >>AB04
  AAF6  NON-NUMERIC, HOW BOUT "A" THRU
  AAFA  "F"
  AAFC  YES! >>AB02
  AAFE  ---
  AAFF  NO, GET OUT NOW
  AB01  RETURN
  AB02  "A" THRU "F", CONVERT TO $BA-$BF
  AB04  ISOLATE OIGIT
  AB08  SHIFT ACCUM 4 BITS LEFT TO MAKE ROOM <AB17>
  AB0B  (WATCH OUT FOR OVERFLOW) >>AAEA
  AB10  OR IN NEW NIBBLE (BCAF)
  AB13  AND REPLACE IN ACCUM LSB (BCAF)
  AB16   OONE

AB17 *********** SHIFT 3 BYTE ACCUM LEFT A BIT **********************
```

```
BASIC Interpreter (BI) -- V1.0.1 -- 1 JAN 84    NEXT OBJECT ADDR: AB17
-------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------

  AB17  SHIFT THE THREE BYTE WORK ACCUM (BCAF)
  AB20  RETURN

AB21 *********** SCAN CMD TABLE FOR COMMAND *************************

  AB21  START WITH LAST COMMAND IN TABLE
  AB26  IS IT A "-" COMMAND? (BCBD)
  AB2B  NOPE >>AB35
  AB2D  YES, SPECIAL COMMAND NUMBER  (BE53)
  AB30  ZERO LENGTH COMMAND STRING (BE52)
  AB33  CONTINUE >>AB52
  AB35  FIRST COMMANDS IN TABLE ARE 8 CHARS
  AB3A  GET INDEX TO NEXT NAME (B8B2)
  AB3D  SAME LENGTH AS LAST NAME? >>AB45
  AB3F  NO,
  AB42  NAMES ARE ONE BYTE SHORTER FROM NOW ON (BE52)
  AB45  ---
  AB46  COMPARE HIS NAME TO MY TABLE (BCBD)
  AB4C  NOT IT... >>AB65
  AB50  COMPARE ENTIRE NAME >>AB46

  AB52  FOUND IT! GET COMMAND INDEX (BE53)
  AB55  *2 FOR MOST THINGS
  AB57  PICK UP PERMITTED PARMS BITS (B97E)
  AB63   EXIT HAPPILY
  AB64  RETURN

  AB65  NOT THE ONE, SKIP TO NEXT (BE52)
  AB6E  IF THERE ARE ANY MORE >>AB3A
  AB70  ELSE, NO SUCH COMMAND (BE53)
  AB74  XRETURN THRU $BE06 VECTOR >>BE06

AB77 *********** ZERO THREE BYTE ACCUM ******************************

  AB77  ZERO THE THREE BYTE WORK
  AB79  ...ACCUMULATOR (BCAF)
  AB82  RETURN

AB83 *********** "-" COMMAND ****************************************

  AB83  CHECK FILE TYPE (BEB8)
  AB86  APPLESOFT PROGRAM?
  AB88  YES, "RUN" IT >>ABF2
  AB8A  BINARY FILE?
  AB8C  YES, "BRUN" IT >>ABCD
  AB8E  TEXT FILE?
  AB90  NO >>AB95
  AB92  YES, "EXEC" IT >>B27B
  AB95  SYS FILE?
  AB97  YES, GO RUN IT >>AB9D
```

```
BASIC Interpreter (BI) -- V1.0.1 -- 1 JAN 84    NEXT OBJECT ADDR: AB99
-------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------

  AB99  ELSE, "FILE TYPE MISMATCH"
  AB9C  RETURN

        ******** RUN "SYS" FILE ****************

  AB9D  CLOSE ALL OPEN FILES <B54C>
  ABA0  CLOSE EXEC <B355>
  ABA5  LSB OF A$ IS 00 (BE58)
  ABA8  FREE UP ALL OF BI'S MEMORY (BF6B)
  ABBB  A$2000 IS WHERE IT WILL LOAD (BE59)
  ABC0  TYPE IS "SYS" (BE6A)
  ABCA  FORCE, T, PATHNAME1, AD PARMS (BE56)
  ABCD  GO DO A STANDARD BRUN >>AE5B

ABD0 *********** "CHAIN" COMMAND ***************************************

  ABD0  SQUASH VARIABLES UP AGAINST HIMEM <A449>
  ABD5  SAVE HIMEM (BC7B)
  ABDC  SET NEW HIMEM BELOW COMBINED VARS
  ABDE  LOAD FILE (LEAVE OTHERS OPEN) <AC47>
  ABE4  RESTORE OLD HIMEM
  ABE6  ERROR? >>AC58
  ABE8  NO, CLEAR VARIABLES <D665>
  ABEB  REEXPAND VARIABLES DOWN AGAINST LOMEM <A4AF>
  ABF0  THEN GO "RUN" PROGRAM >>AC07

ABF2 *********** "RUN" COMMAND ****************************************

  ABF2  NO INPUT FILE ACTIVE NOW
  ABF7  NO APPLESOFT ERROR NUMBER
  ABFC  GOT PATHNAME1?
  ABFD  NO, ERROR >>AC19
  ABFF  YES, LOAD PROGRAM <AC42>
  AC02  ERROR? >>AC58
  AC04  NO, CLEAR VARIABLES <D665>

  AC07  CLEAR ERROR FLAG
  AC09  POSITION TO LINE NUMBER IF GIVEN <ACDC>
  AC0C  RESTORE MY INTERCEPTS <9A8D>
  AC0F  CLEAR COMMAND NUMBER ETC., MODE = 4 <AC19>
  AC16  JUMP INTO APPLESOFT TO RUN PROGRAM >>D7D2

AC19 *********** CLEAR COMMAND NUMBER ETC. ***************************

  AC19  SET NORMAL (NON-INVERSE OR FLASH) <F273>
  AC1E  SEARCH CHARACTER FOR TRACE IS "#" (9F98)
  AC23  NO COMMAND NUMBER NOW (BE53)
  AC26  NO PROMPT
  AC2A  SET MODE=4 (DEFERRED) <9FAD>
  AC2D  "SYNTAX ERROR" IF THINGS GO WRONG >>A879
```

```
BASIC Interpreter (BI) -- V1.0.1 -- 1 JAN 84    NEXT OBJECT ADDR: AC2D
-----------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-----------------------------------------------------------------

AC30 *********** "LOAD" COMMANO ***************************************

  AC30  LOAD PROGRAM <AC42>
  AC33  ERROR? IF NOT, FALL THRU TO WARMSTART >>AC58

AC35 *********** WARMDOS: WARMSTART BI ********************************

  AC35  CLEAR APPLESOFT, RESET POINTERS <0665>
  AC38  RESET MODE/SET INTERCEPTS <9A17>
  AC3O  CURSOR HORIZ. = 0 (START OF LINE)
  AC3F  GO WARMSTART APPLESOFT >>D43F

AC42 *********** LOAD A PROGRAM ***************************************

  AC42  CLOSE ALL OPEN FILES <B54C>
  AC45  ERROR? >>AC58
  AC47  GO LOAD FILE <AC59>
  AC4A  ERROR? >>AC58
  AC4C  SET LOMEM = ARRAYS = FREESTART
  AC4E  ALL TO END OF PROGRAM LOADED
  AC58  RETURN

AC59 *********** REAO A PROGRAM FROM A FILE ***************************

  AC59  READ REQUESTED
  AC5B  TYPE = BAS ASSUMED
  AC5D  OPEN THE FILE <B1EE>
  AC60  ERROR? >>AC58
  AC64  MLI: GET EOF <BE70>
  AC67  ERROR? >>AC58
  AC6B  APPLESOFT PROGRAM START --> REAO DATA (BED7)
  AC6E  AOD TO THAT THE EOF MARK TO ... (BEC8)
  AC71  SET AD PARM --> END OF PROGRAM IMAGE (BE58)
  AC7F  OVERFLOW? >>AC83
  AC81  NO, WOULO PROGRAM EXCEED HIMEM?
  AC83  IF SO...
  AC85  "PROGRAM TOO LARGE"  >>AC58
  AC87  ELSE, PICK UP LENGTH AGAIN (BEC8)
  AC8D  ANO GO READ IT IN <B000>
  AC90  ERROR? >>AC58
  AC92  CLOSE FILE <AFFC>
  AC95  ERROR? >>AC58
  AC97  RELOCATE PROGRAM IF NECESSARY <ACA5>
  ACA0  COPY AO PARM TO APPLESOFT PGM END PTR
  ACA4  RETURN
```

```
BASIC Interpreter (BI) -- V1.0.1 -- 1 JAN 84    NEXT OBJECT AODR: ACA4
-----------------------------------------------------------------
AODR    DESCRIPTION/CONTENTS
-----------------------------------------------------------------

ACA5 *********** RELOCATE APPLESOFT PROGRAM ***************************

  ACA5  ---
  ACA6  WAS APPLESOFT PROGRAM SAVED FROM SAME
  ACA8  MEMORY LOCATION? (BEB9)
  ACB7  YES, NOTHING TO OO THEN >>ACOB
  ACBO  ELSE, LOOP THROUGH PROGRAM
  ACBF  ADJUSTING ALL AODRESSES TO
  ACC1  THE NEW LOAD LOCATION
  ACOB  RETURN

ACDC *********** POSITION TO LINE NUMBER ******************************

  ACOC  WAS A LINE NUMBER PARM GIVEN? (BE57)
  ACE2  NO, NEVER MINO >>ACDB
  ACE4  COPY L KEYWORD VALUE TO APPLESOFT'S LINE # (BE68)
  ACEE  THEN CALL APPLESOFT TO FIND THE LINE <D61A>
  ACF4  SUBTRACT ONE FROM THE ADORESS
  ACF6  AND POINT APPLESOFT'S GETCHR SUBROUTINE
  ACF8  AT IT (SO NEXT CHAR REAO WILL BE FIRST
  ACFA  CHARACTER ON THE LINE).
  ACFF  RETURN

AD00 *********** "SAVE" COMMANO ***************************************

  A000  OOES FILE EXIST ALREAOY? >>AD24
  A002  NO, TYPE = BAS
  AD04  IN T KEYWORO VALUE (BE6A)
  A007  ANO MLI LIST (BEB8)
  A00C  ALLOW ALL ACCESSES (READ/WRITE/ETC.) (BEB7)
  A011  SAVE PROGRAM START AODRESS IN (BEA5)
  A014  AUXID'S (BEB9)
  A01F  GO CREATE A NEW FILE <AD8B>
  A022  ERROR? >>AD6D

  AD24  WRITE ACCESS REQUESTED
  A026  BAS TYPE FILE
  A028  OPEN IT <B1EE>
  A02B  ERROR? >>A060
  AD30  SUBTRACT APPLESOFT PTRS TO COMPUTE
  A032   LENGTH OF PROGRAM.
  A033  STORE THIS IN EOF MARK LIST (BEC8)
  A040  MSB OF EOF MARK IS 00 (<64K PGM) (BECA)
  A045  POINT LIST TO PROGRAM AS DATA TO WRITE (BED7)
  A040  WRITE A RANGE TO OISK FILE <B004>
  A050  ERROR? >>A060
  AD54  MLI: SET EOF (TO TRUNCATE OLO LONGER FILE) <BE70>
  AD57  ERROR? >>A06D
  A059  CLOSE THE FILE <AFFC>
  A05C  ERROR? >>A060
```

```
BASIC Interpreter (BI) -- V1.0.1 -- 1 JAN 84     NEXT OBJECT ADDR: AD60
-------------------------------------------------------------------------
ADDR     DESCRIPTION/CONTENTS
-------------------------------------------------------------------------

  AD60  DOES PROGRAM START MATCH AUXID IN FILE INFO?
  AD65  NO, CHANGE IT >>AD6E
  AD6D   ELSE, EXIT

  AD6E  TO CHANGE IT,  (BEB9)
  AD74  EXIT THRU SET FILE INFO ROUTINE >>B833

AD77 *********** "CREATE" COMMAND ************************************

  AD77  AUXID = 0 (A$ OR RECLN)
  AD82  TYPE KEYWORD GIVEN?
  AD84  YES >>AD8B
  AD88  NO, ASSUME TYPE = DIR (BE6A)

  AD8B  *** CREATE FILE ENTRY *** (BE43)
  AD8E  EXEC FILE ACTIVE?
  AD91  HOW MANY FILES ARE OPEN INCLUDING EXEC? (BE4D)
  AD94  8 OR MORE?
  AD96  YES, ERROR >>ADB3
  AD9B  ELSE, SET TYPE IN MLI LIST (BEA4)
  AD9E  FULL ACCESS (READ/WRITE/ETC.)
  ADA0  KIND = STANDARD FILE
  ADA2  DIR FILE WANTED?
  ADA4  NO >>ADA8
  ADA6  YES, KIND = DIR FILE
  ADA8  SET ACCESS (BEA3)
  ADAB  AND KIND (BEA7)
  ADB0  MLI: CREATE (DON'T COME BACK HERE) >>BE70

  ADB3  "RAM TOO LARGE" ERROR
  ADB5  RETURN

ADB6 *********** "RENAME" COMMAND ************************************

  ADB6  ---
  ADBA  SECOND PATHNAME GIVEN?
  ADBD  IF SO, GO MLI: RENAME >>ADC4
  ADBF  "SYNTAX ERROR" OTHERWISE >>A879

ADC2 *********** "DELETE" COMMAND ************************************

  ADC2  SETUP MLI: DELETE CALL TYPE
  ADC4  EXIT THRU MLI CALL >>BE70

ADC7 *********** "LOCK" COMMAND **************************************

  ADC7  GET FILE INFO FOR PATHNAME1 <B82A>
  ADCA  GET ACCESS CODES (BEB7)
  ADCD  TURN OFF ALL...
  ADCF  BUT READ
  ADD4  THEN GO SET UPDATED FILE INFO >>B841
```

```
BASIC Interpreter (BI) -- V1.0.1 -- 1 JAN 84     NEXT OBJECT ADDR: ADD4
-------------------------------------------------------------------------
ADDR     DESCRIPTION/CONTENTS
-------------------------------------------------------------------------

ADD7 *********** "UNLOCK" COMMAND **********************************

  ADD7  GET FILE INFO FOR PATHNAME1 <B82A>
  ADDA  TURN ON ALL FILE ACCESSES
  ADE2  THEN GO SET UPDATED FILE INFO >>B841

ADE5 *********** "PREFIX" COMMAND ************************************

  ADE5  SLOT/DRIVE GIVEN ON COMMAND? (BE57)
  ADEB  IF SO, GOT OPERAND ALREADY >>ADF1
  ADED  ELSE, (BE56)
  ADF0  CHECK FOR PATHNAME1
  ADF1  AND GO DO MLI: SET PREFIX ...
  ADF3  IF IT'S THERE >>ADC4
  ADF5  ELSE, IS BASIC PROGRAM RUNNING?
  ADF7  IF SO, SET PREFIX ACTIVE FLAG >>AE16
  ADF9  NO, NEW LINE <9FE2>
  AE01  END OF NAME YET? >>AE0E
  AE03  NO, COPY NAME IN PATHNAME1 BUFFER (BCBD)
  AE08  TO OUTPUT DEVICE <9FE4>
  AE0E  AND SKIP A BLANK LINE <9FE2>
  AE15   DONE

  AE16  SET PREFIX ACTIVE FLAG
  AE18  SO BASIC CAN READ THE PREFIX (BE46)
  AE1C  RETURN

AE1D *********** "BSAVE" COMMAND ************************************

  AE1D  PATHNAME1 FOUND? >>AE53
  AE1F  NO, NEW FILE (BE57)
  AE22  AD, L, AND E POSSIBLE
  AE24  AD AND EITHER L OR E REQUIRED
  AE26  OR ELSE ERROR >>AE57
  AE2B  PUT AD IN CREATE PARAMETER LIST (BEA5)
  AE2E  AND IN GET FILE INFO LIST (BEB9)
  AE3C  TYPE = BIN ASSUMED (BE6A)
  AE45  T KEYWORD GIVEN?
  AE47  IF SO, ERROR >>AE57
  AE49  GO CREATE THE FILE <AD8B>
  AE4C  ERROR? >>AE59
  AE4E  GET FILE INFO <B82A>
  AE51  ERROR?  >>AE59
  AE53  WRITING...
  AE55  GO PROCESS LIKE A BLOAD OTHERWISE >>AE6A
```

```
BASIC Interpreter (BI) -- V1.0.1 -- 1 JAN 84    NEXT OBJECT ADDR: AE55
-------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------

  AE57  "PATH NOT FOUND" ERROR
  AE59  ---
  AE5A  RETURN

AE5B ********** "BRUN" COMMAND ***************************************
          (DOES NOT SET MODE=4 SO DOS COMMANDS MAY
          NOT BE ISSUED AS WITH A BASIC PROGRAM)

  AE5B  BLOAD IT FIRST <AE68>
  AE5E  ERROR? >>AE59
  AE60  THEN CALL IT <AE65>
  AE63   THEN EXIT
  AE64  RETURN
  AE65  INDIRECT JMP TO BINARY PROGRAM >>BED7

AE68 ********** "BLOAD" COMMAND **************************************

  AE68  READING...
  AE6A  TYPE = BIN
  AE6C  OPEN THE FILE <B1EE>
  AE6F  ERROR? >>AE59
  AE71  ASSUME USER SPECIFIED AD KEYWORD (BE58)
  AE7A  IF SO, USE HIS ADDRESS >>AE8C
  AE7C  ELSE, USE AD IN FILE INFO AUXID (BEB9)
  AE85  WAS T KEYWORD GIVEN?
  AE87  YES, INVALID PARM (ONLY BIN IS LEGAL) >>AEC3
  AE8C  POINT READ/WRITE PARMS TO DATA (BED7)
  AE92  AND SAVE THIS ADDRESS IN AUXID (BEB9)
  AE98  PICK UP LENGTH FROM L KEYWORD VALUE (BE5F)
  AE9E  WAS L OR E GIVEN?
  AEA0  NEITHER >>AEC7
  AEA2  BOTH?
  AEA4  YES...NAUGHTY! >>AEC3
  AEA6  E GIVEN?
  AEA8  NO, MUST BE L >>AEDD
  AEAA  YES... (BE5D)
  AEAE  COMPUTE L = (E - AD) (BE58)
  AEBA  PLUS ONE FOR INCLUSIVE RANGE >>AEBD
  AEBD  MAKE SURE NO BORROW OCCURED >>AEDD

  AEBF  OR ELSE, "RANGE ERROR"
  AEC2  RETURN

  AEC3  "INVALID PARM" ERROR
  AEC6  RETURN
```

```
BASIC Interpreter (BI) -- V1.0.1 -- 1 JAN 84    NEXT OBJECT ADDR: AEC6
-------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------

  AEC7  ---
  AEC9  MLI: GET EOF <BE70>
  AECC  ERROR? >>AEDB
  AECE  GET L (EOF MARK) (BEC8)
  AED4  BETTER NOT EXCEED 64K (BECA)
  AED7  NO.. >>AEDD

  AED9  YES, "PROGRAM TOO LARGE"
  AEDB  ---
  AEDC  RETURN

  AEDD  STORE LENGTH TO READ OR WRITE (BED9)
  AEE6  B KEYWORD GIVEN?
  AEE8  NO >>AF0F
  AEEC  YES, COPY B VALUE TO SET MARK LIST (BE5A)
  AEF5  ---
  AEF7  MLI: SET MARK <BE70>
  AEFD  NO ERROR? >>AF0F
  AEFF  ERROR, RANGE ERROR?
  AF01  NO >>AEDB
  AF03  BSAVING (NOT BLOAD/BRUNING)?
  AF05  NO >>AEDB
  AF09  MLI: FORCE EOF FORWARD TO MARK <BE70>
  AF0C  AND TRY SET MARK AGAIN >>AEF5
  AF0E  RETURN
  AF0F  GET COMMAND NUMBER (BE53)
  AF12  ASSUME READ
  AF14  BSAVE?
  AF16  NO, READ IS CORRECT >>AF32
  AF18  ELSE, BSAVING... (BE57)
  AF1B  L OR E GIVEN?
  AF1D  NO, RESAVING, GO RIGHT NOW >>AF30
  AF22  MUST UPDATE EOF TO NEW PLACE (BEC8)
  AF2D  MLI: SET EOF <BE70>
  AF30  WRITING
  AF32  MLI: READ OR WRITE <BE70>
  AF35  ERROR? >>AEDB
  AF37  NO, BSAVE?
  AF39  NO >>AF3E
  AF3B  YES, SET FILE INFO WITH AD AND L VALUES <B833>
  AF3E  THEN EXIT THRU CLOSE  >>AFFC

AF41 ********** "STORE" COMMAND **************************************

  AF41  PATHNAME1 EXISTS? >>AF55
  AF43  NO, T = VAR BY DEFAULT
  AF4B  FULL ACCESS (READ/WRITE/ETC.)
  AF50  CREATE THE FILE <AD8B>
  AF53  ERROR? >>AFA1
  AF55  COMPRESS APPLESOFT VARS AGAINST HIMEM <A449>
```

```
BASIC Interpreter (BI) -- V1.0.1 -- 1 JAN 84    NEXT OBJECT ADDR: AF5C
------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
------------------------------------------------------------------

  AF5C  OPEN "VAR" FILE FOR WRITE <B1EE>
  AF5F  ERROR? >>AF9A
  AF61  POINT TO INTERNAL 5 BYTE HEADER BUFFER <AFA2>
  AF64  AND WRITE OUT LENGTHS OF VARS <B004>
  AF67  ERROR? >>AF9A
  AF69  STORE ADDRESS OF VARS (BC8E)
  AF6C  IN READ/WRITE PARM LIST (BED7)
  AF6F  AND FILE INFO AUXID (BEB9)
  AF7B  GET LENGTH OF VARS (BC91)
  AF81  AND WRITE THEM OUT <B004>
  AF84  ERROR? >>AF9A
  AF88  MLI: GET MARK <BE70>
  AF8D  MLI: SET NEW EOF (TRUNCATE IF NECESSARY) <BE70>
  AF90  ERROR? >>AF9A
  AF92  SET FILE INFO WITH AD OF VARS <B833>
  AF95  ERROR? >>AF9A
  AF97  CLOSE FILE <AFFC>
  AF9A  ---
  AF9C  REEXPAND VARS BACK AGAIN <A4AF>
  AFA1  RETURN

AFA2 *********** SETUP TO READ/WRITE VAR HDR ************************
          APPLESOFT VARIABLES HEADER CONSISTS OF:
          2 BYTE LENGTH OF SIMPLE+ARRAY VARIABLES
          2 BYTE LENGTH OF SIMPLE VARIABLES ONLY
          1 BYTE MSB OF HIMEM FOR THESE VARIABLES

  AFA2  STORE ADDRESS OF 5 BYTE INFO
  AFA4  IN READ/WRITE PARM LIST (BED7)
  AFAE  LENGTH = 5
  AFB0  RETURN

AFB1 *********** "RESTORE" COMMAND ***********************************

  AFB1  TYPE = VAR
  AFB3  READING
  AFB5  OPEN THE FILE <B1EE>
  AFB8  ERROR? >>AFA1
  AFBA  SET UP TO READ THE HEADER <AFA2>
  AFBD  READ 5 BYTE HEADER <B000>
  AFC0  ERROR? >>AFA1
  AFC2  PICK UP WHERE TO READ IN COMPRESSED VARS (BEB9)
  AFC5  FROM AUXID (BC8E)
  AFCB  ADJUST MSB OF THIS BY THE DIFFERENCE
  AFCE  BETWEEN HIMEM'S (NOW AND WHEN STORED) (BC8D)
  AFDB  MAKE SURE VARS WON'T OVERLAY PROGRAM
  AFDD  IF SO, ERROR >>AFF8
  AFE7  COMPUTE LENGTH OF ALL VARS/STRINGS
  AFE9  (HIMEM-START) (BC8F)
  AFED  GO READ COMBINED VARS INTO MEMORY <B000>
  AFF0  ERROR? >>AFA1
```

```
BASIC Interpreter (BI) -- V1.0.1 -- 1 JAN 84    NEXT OBJECT ADDR: AFF2
------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
------------------------------------------------------------------

  AFF2  CLOSE THE FILE <AFFC>
  AFF5  EXIT BY REEXPANDING THE VARS DOWN >>AF9A

  AFF8  "PROGRAM TOO LARGE" ERROR
  AFFB  RETURN

AFFC *********** CLOSE FILE ***************************************

  AFFC  SET MLI CLOSE OPCODE
  AFFE  AND GO TO MLI >>B00C

B000 *********** READ/WRITE A RANGE **********************************

  B000  READ MLI OPCODE
  B002  JUMP IN >>B006
  B004  WRITE MLI OPCODE
  B006  STORE LENGTH (BEDA)
  B00C  EXIT THRU MLI:READ OR WRITE >>BE70

B00F *********** "PR#" COMMAND ***************************************

  B00F  USE CSWL AND OUTVEC
  B014  JUMP TO COMMON CODE >>B01D

B016 *********** "IN#" COMMAND ***************************************

  B016  USE KSWL
  B01B  AND INVEC

  B01D  OR IN SLOT GIVEN BY USER (BE6B)
  B020  *2 FOR USE AS INDEX INTO TABLE
  B025  WAS SLOT PARAMETER GIVEN?
  B027  NO... >>B03A
  B029  YES, (BE57)
  B02C  AD GIVEN? >>B04F
  B02E  NO, GET INVEC OR OUTVEC FOR THIS SLOT (BE10)
  B031  AND STORE ON AD KEYWORD VALUE (BE58)
  B03A  VALIDITY CHECK I/O DRIVER <B061>
  B03D  NO GOOD? >>B04E
  B03F  GET INDEX TO CSWL OR KSWL (BCA9)
  B045  AND REPLACE ONE OR THE OTHER WITH (0036)
  B048  HIS ADDRESS (BE59)
  B04E  RETURN

  B04F  VALIDITY CHECK AD KEYWORD VALUE <B061>
  B052  NO GOOD? >>B060
  B054  GOOD, COPY VALUE TO INVEC OR OUTVEC (BE59)
  B060   EXIT BUT DON'T REDIRECT I/O NOW
```

```
BASIC Interpreter (BI) -- V1.0.1 -- 1 JAN 84    NEXT OBJECT ADDR: B060
-------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------

B061 *********** VALIDITY CHECK I/O DRIVER ***************************

  B061  $3A/3B --> NEW HANDLER (FROM AD PARM) (BE58)
  B06D  IS DRIVER IN MAIN RAM (BELOW $C000)?
  B06F  YES >>B086
  B071  NO, RESET I/O CARD ROMS (CFFF)
  B074  USE $3C TO COUNT ITERATIONS
  B076  TEST ROM AT USER'S ADDRESS
  B07C  FOR STABILITY
  B080  256 TIMES
  B084   MUST BE OK
  B085  RETURN
  B086  MAIN RAM I/O DRIVER
  B088  MUST START WITH A "CLD" INSTRUCTION
  B08A  OK... >>B084

  B08C  ELSE, "NO DEVICE CONNECTED"
  B08F  RETURN

B090 *********** "CAT" COMMAND ***************************************

  B090  39 CHARACTERS PER LINE
  B092  THEN PROCESS LIKE "CATALOG" >>B096

B094 *********** "CATALOG" COMMAND ***********************************

  B094  79 CHARACTERS PER LINE
  B096  STORE LINE LENGTH (BCB6)
  B09C  TEST FOR T AND
  B09E  ....PATHNAME1 GIVEN
  B09F  GOT T >>B0A4
  B0A1  NO T, T=0 (ANY TYPE WILL DO) (BE6A)
  B0A4  GOT PATHNAME1 >>B0AB
  B0A6  NO PATHNAME1, GET FILE INFO FOR PREFIX <B82A>
  B0A9  ERROR? >>B111
  B0AB  OPEN/READ DIRECTORY HEADER <B1A4>
  B0AE  ERROR? >>B111
  B0B0  SKIP TO A NEW LINE <9FE2>
  B0B3  FORMAT DIRECTORY'S NAME TO $201 <B112>
  B0B6  PRINT $201 <9FD4>
  B0B9  SKIP TO A NEW LINE <9FE2>
  B0BC  BLANK $201 BUFFER <A6A9>
  B0C1  UNPACK HEADING MESSAGE LINE <9FE7>
  B0C4  PRINT IT (40 OR 80 COLUMNS) <9FD4>
  B0C7  SKIP TO A NEW LINE <9FE2>
  B0CD  ANY FILES IN THIS DIRECTORY? (BCBA)
  B0D0  NO >>B0FD
  B0D2  YES, READ NEXT ENTRY <B22B>
  B0D5  ERROR? >>B111
  B0D7  GET TYPE REQUESTED FOR SEARCH (BE6A)
```

```
BASIC Interpreter (BI) -- V1.0.1 -- 1 JAN 84    NEXT OBJECT ADDR: B0DA
-------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------

  B0DA  ANY TYPE WILL DO? >>B0E1
  B0DC  NO, CHECK TYPE AGAINST THIS ENTRY (0269)
  B0DF  NOT IT, SKIP IT >>B0E7
  B0E1  ELSE, FORMAT ENTRY TO $201 <A501>
  B0E4  AND PRINT $201 <9FD4>
  B0E7  CHECK KEYBOARD (C000)
  B0EA  FOR A CONTROL-C
  B0EC  IGNORE ANYTHING ELSE >>B0F8

  B0EE  CONTROL-C, WHAT STATE ARE WE IN? (BE42)
  B0F1  DEFERRED >>B0FD
  B0F3  NO, IMMEDIATE, RESET KEYBOARD STROBE (C010)
  B0F6  AND EXIT RIGHT NOW >>B0FD

  B0F8  ELSE, ANY FILES LEFT IN COUNT? (BCBA)
  B0FB  YES, CONTINUE >>B0D2
  B0FD  ELSE, CLOSE DIRECTORY <AFFC>
  B100  ERROR? >>B111
  B102  SKIP TO A NEW LINE <9FE2>
  B105  FORMAT BLOCKS FREE AND IN USE TO $201 <B141>
  B108  ERROR? >>B111
  B10A  PRINT $201 <9FD4>
  B10D  SKIP A LINE <9FE2>
  B111   DONE

B112 *********** FORMAT NAME OF DIRECTORY ****************************

  B112  BLANK $201 BUFFER <A6A9>
  B115  FILE NAME IS AT +1 INTO DIR ENTRY
  B117  GET NAME LENGTH/TYPE (025D)
  B11C  VOLUME DIRECTORY HEADER?
  B11E  NO >>B124
  B120  YES, START NAME WITH "/" (0200)
  B124  ---
  B125  ISOLATE NAME LENGTH FROM TYPE
  B127  AND SET UP LENGTH TO COPY (0200)
  B12C  COPY DIRECTORY NAME TO (0259)
  B131  ...LINE (0200)
  B13B  SET $200 TO MAXIMUM LENGTH
  B140  RETURN

B141 *********** FORMAT BLOCKS FREE/INUSE ****************************

  B141  POINT MLI:ONLINE PARMLIST
  B143  TO TXTBUF (PATHNAME1) (BEC8)
  B14B  COPY DEVICE NUMBER (UNIT) (BF30)
  B153  MLI: ONLINE <BE70>
  B156  ERROR? >>B111
  B15B  ISOLATE NAME LENGTH FROM BUFFER
  B15E   BUMP BY ONE TO INCLUDE "/"
  B15F  AND STORE IT AS A PREFIX (BCBC)
```

```
BASIC Interpreter (BI) -- V1.0.1 -- 1 JAN 84    NEXT OBJECT ADDR: B164
------------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
------------------------------------------------------------------------

  B164  STORE "/" AS FIRST CHARACTER (BCBD)
  B167  GET FILE INFO FOR PREFIX <B82A>
  B16A  ERROR? >>B111
  B16C  BLANK $201 BUFFER <A6A9>
  B171  UNPACK "BLOCKS FREE: BLOCKS USED.." <9FE7>
  B174  ZERO THE THREE BYTE ACCUM <AB77>
  B17F  CONVERT AUXID (TOTAL BLOCKS) <A66C>
  B18A  CONVERT BLOCKS USED <A66C>
  B191  BLOCKS_FREE = TOTAL_BLOCKS (BEBC)
  B198  ... - BLOCKS_USED (BEBD)
  B19F  CONVERT BLOCKS FREE <A66C>
  B1A3   DONE!

B1A4 *********** OPEN/READ DIRECTORY HDR ****************************

  B1A4  READ ONLY
  B1A8  CHECK FILE KIND (BEBB)
  B1AB  VOLUME DIRECTORY?
  B1AD  NO >>B1B2
  B1AF  YES, TYPE = DIR (BEB8)
  B1B2  OPEN THE FILE <B1FA>
  B1B5  ERROR? IF NOT, FALL THRU >>B1ED

B1B7 *********** READ DIRECTORY HDR **********************************

  B1B7  BUFFER IS $259
  B1C3  LENGTH IS $2B (ONE ENTRY) (BED9)
  B1CD  MLI: READ <BE70>
  B1D0  ERROR? >>B1ED
  B1D4  COPY ENTRY LENGTH, ENTRIES PER BLOCK, (027C)
  B1D7  AND FILE COUNT FROM DIR HDR (BCB7)
  B1DD  STORE ENTRY LENGTH IN READ LENGTH NOW (BED9)
  B1E2  SET COUNTER TO FIRST ENTRY IN BLOCK (BCBB)
  B1E7  MARK = 0 (START OF FILE) (BEC9)
  B1ED  RETURN

B1EE *********** OPEN FILE ******************************************
          A REGISTER = ACCESS BITS
          X REGISTER = DEFAULT TYPE

  B1EE  ---
  B1F2  T KEYWORD GIVEN?
  B1F4  NO >>B1F9
  B1F6  YES, USE KEYWORD VALUE INSTEAD (BE6A)
  B1F9  ---
  B1FA  EXISTING FILE OF THIS TYPE? (BEB8)
  B1FD  NO, ERROR  >>B223
  B1FF  CHECK ACCESS REQUESTED (BEB7)
  B202  REQUESTED ACCESS NOT PERMITTED >>B227
  B204  SET SYSTEM BUFFER IN OPEN PARM LIST (BC88)
  B20C  LEVEL = $0F (BF94)
```

```
BASIC Interpreter (BI) -- V1.0.1 -- 1 JAN 84    NEXT OBJECT ADDR: B211
------------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
------------------------------------------------------------------------

  B211  MLI: OPEN <BE70>
  B214  ERROR? >>B222
  B219  SAVE REFNUM IN READ/WRITE PARMLIST (BED6)
  B21C  AND CLOSE PARMLIST (BEDE)
  B21F  AND GET/SET EOF/MARK LIST (BEC7)
  B222   AND EXIT

  B223  "FILE TYPE MISMATCH"
  B226  RETURN
  B227  "FILE LOCKED"
  B22A  RETURN

B22B *********** READ NEXT DIRECTORY ENTRY ***************************

  B22B  FORCE MARK TO START OF THIS BLOCK (BEC9)
  B233  CHECK ENTRY NUMBER (BCBB)
  B238  LAST ENTRY IN THIS BLOCK? (BCB8)
  B23B  NO >>B247
  B23E  YES, ENTRY 0 NEXT TIME (BCBB)
  B241  BUMP MARK TO NEXT BLOCK (BEC9)
  B247  ---
  B249  MARK POSITIONED TO PROPER ENTRY YET? >>B252
  B24B  NO, BUMP POINTER TO NEXT ENTRY (BCB7)
  B24E  AND CONTINUE IF STILL FIRST PAGE >>B247
  B250  JUST ENTERED SECOND PAGE >>B244
  B252  ADD 4 TO PTR TO ADJUST FOR BLOCK PREFIX
  B259  MLI: SET MARK <BE70>
  B25C  ERROR? >>B277
  B260  MLI: READ <BE70>
  B263  ERROR? >>B277
  B265  BUMP ENTRY COUNTER (BCBB)
  B26B  IS THIS ENTRY VALID?
  B26D  NO, SKIP OVER IT >>B22B
  B26F  DECREMENT FILE COUNT (BCB9)
  B277   AND RETURN TO CALLER

B278 *********** EXTERNAL COMMAND HANDLER ****************************

  B278  INDIRECT JMP TO XTRNAD VECTOR >>BE50

B27B *********** "EXEC" COMMAND ***************************************

  B27B  IS THIS FILE OPEN ALREADY? <B479>
  B27E  NO >>B2AA
  B280  YES, EXEC CLOSING? (BE4E)
  B283  NO >>B2A6
  B285  SAVE REFNUM (BEC7)
  B28A  RESET MARK TO ZERO (BEC8)
  B295  MLI: SET MARK <BE70>
  B298  ERROR? >>B29F
  B29A  GET REFNUM AGAIN (BEC7)
```

```
BASIC Interpreter (BI) -- V1.0.1 -- 1 JAN 84    NEXT OBJECT ADDR: B29D
-------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------

 B29D  GO RESTART THIS EXEC FILE FROM ITS START >>B31D

        ******** CLOSE EXEC FILE ****************

 B29F   PRESERVE CALLER'S AREG
 B2A0  AND CLOSE THE FILE <B355>
 B2A5   THEN RETURN WITH ERROR

 B2A6  "FILE BUSY" ERROR
 B2A9  RETURN

        ******** CONTINUE EXEC SETUP ************

 B2AA  EXEC ACTIVE? (BE43)
 B2AD  NO >>B2B4
 B2AF  YES, CLOSE IT <B355>
 B2B2  ERROR? >>B2BD
 B2B4  GET FILE TYPE (BEB8)
 B2B7  SHOULD BE TXT
 B2B9  IT IS >>B2BF

 B2BB  ELSE, "FILE TYPE MISMATCH"
 B2BD   RETURN WITH ERROR
 B2BE  RETURN

 B2BF  MOVE STRINGS TO MAKE ROOM FOR A BUFFER <A232>
 B2C2  NO ROOM? >>B2BD
 B2C6  STORE NEW BUFFER ADDRESS IN PARM LIST (BEC8)
 B2CF  GET COUNT OF OPEN FILES (BE4D)
 B2D2  NO OTHERS CURRENTLY OPEN? >>B2F8

        ******** MAKE EXEC TOPMOST BUFFER *******

 B2D4   OTHERS ARE OPEN...
 B2D6  OPENCOUNT*4 (4 PAGES PER BUFFER)
 B2D8  ADD THIS TO MY BUFFER TO FIND TOP BUFFER (BC88)
 B2DC  SEARCH OPEN FILES TO FIND THE FILE WHICH (BC93)
 B2DF  IS USING THIS BUFFER... >>B2E5
 B2E4   IF IT IS NOT FOUND, BREAK!
 B2E5  ---
 B2E6  MOVE THAT FILE TO THE NEW BUFFER INSTEAD (BC93)
 B2E9  GET THAT FILE'S REFNUM ALSO (BC9B)
 B2F1  MLI: SET BUFF <BE70>
 B2F4  NO ERRORS? >>B2F7
 B2F6   IF ERROR, BREAK!
 B2F7  ---
```

```
BASIC Interpreter (BI) -- V1.0.1 -- 1 JAN 84    NEXT OBJECT ADDR: B2F7
-------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------

        ******** OPEN NEW EXEC FILE *************

 B2F8  SET NEW BUFFER ALLOCATION PAGE (BC88)
 B2FB  SET UP OPEN LIST FOR EXEC TOO (BECF)
 B300  LEVEL = 0 (BF94)
 B305  MLI: OPEN (EXEC FILE) <BE70>
 B308  NO ERROR? >>B311

 B30A  ---
 B30B  IF ERROR, FREE BUFFER FIRST <A289>
 B310   THEN EXIT WITH ERROR

 B311  SAVE BUFFNO FOR EXEC (BECF)
 B317  AND REFNUM TOO (BED0)

        ******** COMPLETE EXEC COMMAND **********

 B31D  SAVE READ REFNUM (BED6)
 B320  AND GET/SET REFNUM (BEC7)
 B323  AND NEWLINE REFNUM (BED2)
 B329  SET "L" VALUE FROM AUXID (BE5F)
 B332  SAVE PATHNAME/AUXID IN OPEN FILE TABLE <B445>
 B337  IGNORE MSB FOR END OF LINE CHARS (BED3)
 B33C  MLI: SET NEWLINE <BE70>
 B342  WAS "F" OR "R" GIVEN ON COMMAND LINE?
 B344  NO >>B34E
 B346  YES, POSITION TO SPECIFIED STARTING PT <B57C>
 B349  NO ERRORS? >>B34E
 B34B  IF ERROR, GO CLOSE EXEC >>B29F
 B34E  MARK EXEC ACTIVE
 B354   AND RETURN TO CALLER

B355 ********** CLOSE EXEC FILE ************************************

 B355  EXEC ACTIVE? (BE43)
 B358  NO, SKIP IT >>B365
 B35A  INDICATE EXEC FILE CLOSING (BE4E)
 B35F  PICK UP REFNUM FOR EXEC (BC9B)
 B362  AND GO CLOSE IT <B4FF>
 B365  RETURN

B366 ********** "VERIFY" COMMAND ************************************

 B366  FILE NOT FOUND? >>B3A1
 B36B  FILE FOUND, WAS A PATHNAME1 GIVEN?
 B36D  YES >>B377
 B36F  NO,
 B371  PRINT "(C) APPLE COMPUTER..." <9FC3>
 B374  AND A NEW LINE <9FE2>
 B377   THEN EXIT
```

```
BASIC Interpreter (BI) -- V1.0.1 -- 1 JAN 84    NEXT OBJECT ADDR: B378
-------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------

  B378  RETURN

B379 *********** FLUSH ALL OPEN FILES *******************************

  B379  REFNUM = 0 (ALL FILES)
  B37B  JUMP INTO FLUSH >>B389

B37D *********** "FLUSH" COMMAND ************************************

  B37D  ---
  B380  WAS PATHNAME1 GIVEN?
  B382  NO, FLUSH ALL FILES >>B389
  B384  ELSE, LOOK UP NAME IN OPEN FILE LISTS <B479>
  B387  NOT AN OPEN FILE >>B391
  B389  SAVE REFNUM IN PARM LIST (BEDE)
  B38E  MLI: FLUSH <BE70>
  B391   EXIT

B392 *********** "OPEN" COMMAND *************************************

  B392  ---
  B393  LOOK UP NAME IN OPEN FILE LIST <B479>
  B396  NOT CURRENTLY OPEN? >>B3A5

  B398  ---
  B399  IT IS OPEN, "FILE BUSY" ERROR
  B39C  RETURN

  B39D  "FILE TYPE MISMATCH" ERROR
  B3A0  RETURN

  B3A1  "PATH NOT FOUND" ERROR
  B3A3  ---
  B3A4  RETURN

  B3A5  ---
  B3A6  ASSUME "L" IS ZERO
  B3AD  WAS "L" KEYWORD GIVEN?
  B3AF  YES, USE HIS VALUE >>B3B7
  B3B1  NO, SET "L" TO ZERO (BE60)
  B3BA  WAS "T" GIVEN?
  B3BE  YES, USE HIS TYPE >>B3C5
  B3C0  ELSE, DEFAULT TO "TXT"
  B3C5  DOES THE FILE ALREADY EXIST? >>B3E8
  B3C7  NO, "T" GIVEN? IF SO, ERROR >>B3A1
  B3C9  FORCE TYPE = "TXT" (BEB8)
  B3CE  FULL ACCESS (BEB7)
  B3D4  COPY "L" KEYWORD VALUE (BE5F)
  B3D7  TO CREATE (BEA6)
  B3DA  AND SET FILE INFO LISTS (BEBA)
  B3E3  GO CREATE THE FILE <AD8B>
```

```
BASIC Interpreter (BI) -- V1.0.1 -- 1 JAN 84    NEXT OBJECT ADDR: B3E6
-------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------

  B3E6  ERROR? >>B3A3
  B3E8  CHECK FILE TYPE (BEB8)
  B3EB  AGAINST HIS "T" VALUE (BE6A)
  B3EE  MISMATCH? >>B39D
  B3F0  NO, TYPE = TXT?
  B3F2  NO >>B407
  B3F4  YES, GET RECORD LENGTH FROM AUXID (BEBA)
  B3FD  WAS "L" KEYWORD VALUE GIVEN?
  B3FF  YES, USE THAT INSTEAD >>B407
  B401  OTHERWISE, SAVE AUXID RECORD LEN (BE60)
  B407  ALLOCATE A NEW FILE BUFFER <A232>
  B40A  ERROR? >>B3A3
  B40C  GET BUFFER PAGE NO. (BC88)
  B40F  AND STORE IN OPEN LIST (BECF)
  B414  LEVEL = 7 (BF94)
  B419  MLI: OPEN <BE70>
  B41C  NO ERRORS? >>B425

  B41E  ---
  B41F  ERROR, FREE BUFFER FIRST <A289>
  B424   THEN EXIT WITH ERROR CODE

  B425  CHECK FILE TYPE AGAIN (BEB8)
  B428  "DIR" FILE?
  B42A  YES >>B42D
  B42C   NO
  B42D  ---
  B430  SET DIR FLAG ACCORDINGLY (BE47)
  B433  USING OPEN COUNT AS AN INDEX (BE4D)
  B439  STORE BUFFER LOCATION IN OPEN FILE LIST (BC94)
  B43F  ALSO, THE REFNUM (BC9C)
  B442  AND BUMP OPEN FILE COUNT AND FALL THRU (BE4D)

B445 *********** SAVE FILE NAME/RECLEN IN TABLE **********************

  B445  MAKE INDEX FROM REFNUM*32 BYTES
  B44B  GET NAME LENGTH (0280)
  B44E  OR IN DIR FLAG (BE47)
  B451  AND STORE IN OPEN FILE NAME LIST (BCFE)
  B457  NAME > OR = TO 30 BYTES?
  B459  NO... >>B45D
  B45B  YES, USE 29
  B45D  STORE THAT AS A LOOP COUNTER
  B462  COPY "L" KEYWORD VALUE TO NAME LIST TOO (BCFF)
  B46B  ---
  B46C  COPY FILE NAME TO NAME LIST (0280)
  B475  COPY ALL OF NAME, THEN FALL THRU TO EXIT >>B46B
```

```
BASIC Interpreter (BI) -- V1.0.1 -- 1 JAN 84    NEXT OBJECT ADDR: B475
-------------------------------------------------------------------
AODR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------

B477 ********** "MON" ANO "NOMON" COMMANDS *************************

  B477 '  IGNORE THESE COMMANDS AND
  B478    RETURN TO CALLER

B479 ********** LOOKUP OPEN FILENAME ********************************
          (RETURNS REFNUM OF OPEN FILE)

  B479  ---
  B47C  WAS PATHNAME1 GIVEN?
  B47E  YES >>B484

  B480  NO, "SYNTAX ERROR"
  B483   EXIT WITH ERROR

  B484  ANY FILES CURRENTLY OPEN? (BE4O)
  B487  NO, CAN'T FIND IT THEN >>B4A2
  B489  YES, CLEAR EXEC FILE CLOSING FLAG (BE4E)
  B48C  STORE FILE COUNT AS LOOP COUNTER
  B48E  GET NEXT REFNUM (BC9B)
  B491  COMPARE FILENAMES  <B4BC>
  B494  NOT THE ONE? >>B49D
  B496  ELSE, WE'VE GOT IT!
  B498  PICK UP APPROPRIATE REFNUM (BC9B)
  B49B  ---
  B49C   ANO RETURN WITH IT
  B49O  ELSE, NOT IT, TRY NEXT ONE
  B4A0  ANO CONTINUE LOOPING >>B48C

  B4A2  CAN'T FINO IT, IS EXEC ACTIVE? (BE43)
  B4A5  NO, THEN WE MUST GIVE UP >>B4B8
  B4AA  IS HE LOOKING FOR EXEC FILE? <B4BC>
  B4AO  NO, GIVE UP >>B4B8
  B4B1  YES, EXEC FILE CLOSING (BE4E)
  B4B6  ANO RETURN WITH EXEC'S REFNUM >>B498

  B4B8  "FILE NOT OPEN" ERROR
  B4BB   RETURN WITH ERROR COOE

B4BC ********** COMPARE FILENAMES **********************************

  B4BC  REFNUM*32 FOR FILENAME INDEX
  B4C2  PICK UP OIR FLAG FROM THIS ENTRY (BCFE)
  B4CA  SAME LENGTH AS HIS FILENAME? (0280)
  B4CD  NO, CAN'T BE IT THEN >>B4F2
  B4O0  MAKE SURE LENGTH DOES NOT EXCEEO 29
  B4O4  IF IT OOES, ONLY LOOK AT FIRST 29
  B4D6  USE $3A AS LOOP COUNTER
  B4DB  COPY "L" OF THIS FILE TO KEYWORO (BCA4)
  B4E4  ---
```

```
BASIC Interpreter (BI) -- V1.0.1 -- 1 JAN 84    NEXT OBJECT ADOR: B4E5
-------------------------------------------------------------------
ADOR    OESCRIPTION/CONTENTS
-------------------------------------------------------------------

  B4E5  COMPARE NAMES (0280)
  B4EB  NO MATCH? EXIT WITH Z FLAG CLEAR >>B4F2
  B4F2   MATCH, EXIT WITH Z FLAG SET

B4F3 ********** "CLOSE" COMMAND **************************************

  B4F3  ---
  B4F6  PATHNAME1 GIVEN?
  B4F8  NO, CLOSE ALL FILES >>B54C
  B4FA  YES, LOOK IT UP IN OPEN FILE TABLES  <B479>
  B4FO  NOT FOUNO? >>B49B
  B4FF  FOUNO IT, STORE REFNUM IN CLOSE LIST (BEOE)
  B505  MARK BUFFER PAGE FREE (BC88)
  B508  EXEC CLOSING? (BE4E)
  B50B  YES...NO NEEO TO COMPRESS LISTS >>B529
  B50D  GET OPEN COUNT (LAST OPENED FILE NO.) (BE4O)
  B511  SWAP BUFFERS (BC93)
  B51F  ANO REFNUMS WITH THE LAST OPENEO FILE (BC9B)
  B529  ---
  B52B  LEVEL = 0 (BF94)
  B530  MLI: CLOSE <BE70>
  B533  ERROR? >>B55C
  B535  RELEASE THE BUFFEk <A289>
  B538  EXEC FILE CLOSING? (BE4E)
  B53B  NO >>B548
  B540  YES, EXEC NO LONGER ACTIVE (BE43)
  B543  ANO NO LONGER CLOSING (BE4E)
  B547   RETURN TO CALLER

  B548  OROP OPEN FILE COUNT (BE4O)
  B54B   ANO EXIT

B54C ********** CLOSE ALL OPEN FILES ********************************

  B54C  ANY FILES OPEN? (BE4O)
  B54F  NO >>B55O
  B551  YES, EXEC NOT CLOSING (BE4E)
  B557  CLOSE LAST FILE OPENEO <B4FF>
  B55A  IF THAT WORKS, START ALL OVER AGAIN >>B54C
  B55C   EXIT WHEN ALL ARE CLOSEO

  B55D  ---
  B55F  SET CLOSE REFNUM TO ZERO (ALL FILES) (BEDE)
  B564  LEVEL = 7 (LEVEL 0 FILES ALREAOY CLOSEO) (BF94)
  B569  EXIT THRU  MLI: CLOSE >>BE70
```

```
BASIC Interpreter (BI) -- V1.0.1 -- 1 JAN 84    NEXT OBJECT ADDR: B56C
-------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------

B56C ********** "POSITION" COMMAND **********************************

  B56C  LOOKUP NAME OF FILE <B479>
  B56F  NOT OPEN? >>B5D9
  B571  SET REFNUM IN READ/WRITE PARMLIST (BED6)
  B574  AND SET NEWLINE LIST (BED2)
  B577  DIR FILE?  (BE47)
  B57A  YES, GET OUT RIGHT NOW! >>B5DA

  B57C  "F" OR "R" GIVEN? (BE57)
  B581  NO, INVALID PARM >>B5D7
  B583  BOTH GIVEN?
  B585  YES, INVALID PARM >>B5D7
  B587  JUST "R" GIVEN?
  B589  NO, JUST "F" >>B597
  B58B  JUST "R", COPY "R" VALUE TO "F" (BE65)
  B58E  ("R" AND "F" ARE ALIASES) (BE63)
  B597  SET COUNT TO 239. (MAXIMUM LINE LEN)
  B5A6  BUFFER IS AT $200 (BED8)
  B5A9  ---
  B5AB  NEW LINE CHAR IS EITHER $0D OR $8D (BED3)
  B5B0  MLI: SET NEWLINE <BE70>
  B5B3  ERROR? >>B5D9

          ******** SKIP LINES BY READING THEM *****

  B5B5  ---
  B5B8  "F" = 0? (BE64)
  B5BC  YES, DONE >>B5DA
  B5BE  ELSE...
  B5C0  MLI: READ NEXT FIELD (LINE) <BE70>
  B5C3  ERROR? >>B5D9
  B5C8  DECREMENT "F" VALUE BY ONE
  B5D5  AND GO CHECK IT AGAIN >>B5B5

  B5D7  "INVALID PARAMETER" ERROR
  B5D9  ---
  B5DA   EXIT TO CALLER

B5DB ********** COMPUTE NEW FILE POSITION ***************************
          (COMPUTES ABSOLUTE FILE POSITION MARK)

  B5DB  ACCUM = CURRENT RECORD LENGTH (BCA4)
  B5EF  MARK = 0 (BEC8)

          ******** MARK = "R" * RECLEN ************
```

```
BASIC Interpreter (BI) -- V1.0.1 -- 1 JAN 84    NEXT OBJECT ADDR: B5F8
-------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------

  B5F8  SHIFT "R" VALUE RIGHT (BE66)
  B600  IF LOW BIT OFF, NO ADD >>B619
  B603  ADD ONE INSTANCE OF RECLEN TO MARK (BCAF)
  B612  OVERFLOW? >>B62C
  B617  ACCUM OVERFLOW? >>B62C
  B619  SCALE ACCUM (MULTIPLIER) UP BY 2 (BCAF)
  B622  IF "R" NON ZERO... (BE65)
  B628  CONTINUE LOOPING >>B5F8
  B62B   ELSE, EXIT TO CALLER

  B62C  "RANGE ERROR"
  B62F  RETURN

B630 ********** "READ" COMMAND ***************************************

  B630  LOOK UP FILE NAME <B479>
  B633  NOT OPEN? >>B685
  B635  ITS OPEN, STORE REFNUM IN READ/WRITE... (BED6)
  B638  GET/SET... (BEC7)
  B63B  AND SET NEWLINE PARMLISTS (BED2)
  B63E  DIR FILE? (BE47)
  B641  YES, SPECIAL HANDLING REQUIRED >>B686
  B643  NO, PRE-POSITION FOR "B", "F", OR "R" <B6C0>
  B646  ERROR POSITIONING? >>B685
  B648  ASSUME "L" = 239.
  B64F  "L" GIVEN?
  B651  NO >>B666
  B653  YES, USE HIS "L" VALUE (BE5F)
  B659  UNLESS ITS >256 >>B6BB
  B65D  OR >239. >>B6BB
  B661  DOUBLE QOUTE IT SO COMMAS COME THRU (0200)
  B664  READ INTO $201
  B666  IF NO "L", READ TO $200 (BED7)
  B66C  NL CHAR = $0D/$8D (OR NONE IF "L")  (BED3)
  B67B  MLI: SET NEWLINE <BE70>
  B67E  ERROR? >>B685
  B680  ---
  B682  MARK INPUT "READ" FILE ACTIVE (BE44)
  B685   AND RETURN

          ******** READ DIR FILE ******************

  B686  SET READ/WRITE LIST REFNUM (BED6)
  B689  AND GET/SET LIST REFNUM (BEC7)
  B68E  READING TO $259 (BED7)
  B698  INIT CAT FLAG TO FIRST LINE VALUE (BE4F)
  B69E  "R" GIVEN?
  B6A1  NO, DONE >>B680
  B6A5  YES, ZERO OUT MARK (BEC8)
  B6B0  MLI: REWIND FILE <BE70>
```

```
BASIC Interpreter (BI) -- V1.0.1 -- 1 JAN 84    NEXT OBJECT ADDR: B6B3
-------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------

  B6B3  ERROR? >>B6BA
  B6B7  MARK INPUT FILE ACTIVE (BE44)
  B6BA   AND EXIT

B6BB ********** "RANGE ERROR" ***************************************

  B6BB  "RANGE ERROR" CODE
  B6BF   EXIT TO CALLER

B6C0 ********** PRE-POSITION FOR I/O ********************************

  B6C0  ---
  B6C3  "B", "F", OR "R" GIVEN?
  B6C5  NO, EXIT >>B709
  B6C7  "R"?
  B6C9  NO >>B6D5
  B6CB  YES, COMPUTE ABSOLUTE POSITION <B5DB>
  B6CE  ERROR? >>B6BB
  B6D0  NO, SET MARK TO NEW POSITION <B702>
  B6D3  ERROR? >>B70A
  B6D5  "F" GIVEN? (BE57)
  B6DA  NO >>B6E1
  B6DC  SKIP LINES UNTIL "F" = 0 <B5A9>
  B6DF  ERROR? >>B70A
  B6E1  "B" GIVEN?  (BE57)
  B6E6  NO >>B709
  B6EA  MLI: GET MARK <BE70>
  B6ED  ERROR? >>B70A
  B6F3  ADD "B" VALUE TO CURRENT MARK (BE5A)
  B6F6  (3 BYTE ADD) (BEC8)
  B700  OVERFLOW? >>B6BB
  B702  ---
  B704  MLI: SET MARK <BE70>
  B707  ERROR? >>B70A
  B709  ---
  B70A  ---
  B70C   EXIT TO CALLER

B70D ********** "WRITE" COMMAND ***************************************

  B70D  LOOKUP OPEN FILE NAME <B479>
  B710  NOT AN OPEN FILE? >>B722
  B712  STORE READ/WRITE REFNUM (BED6)
  B715  AND GET/SET REFNUM (BEC7)
  B718  AND NEWLINE REFNUM IN PARM LISTS (BED2)
  B71B  DIR FILE? (BE47)
  B71E  NO, OK >>B724
```

```
BASIC Interpreter (BI) -- V1.0.1 -- 1 JAN 84    NEXT OBJECT ADDR: B71E
-------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------

  B720  YES, "FILE LOCKED" ERROR
  B722  ---
  B723   EXIT TO CALLER

  B724  DATA BUFFER AT $200
  B72E  PRE-POSITION FOR "B", "F", AND "R" <B6C0>
  B731  NO ERRORS? >>B747
  B733  WAS ERROR A RANGE ERROR?
  B735  NO, REAL ERROR >>B722
  B737  YES, MY RANGE ERROR OR MLI'S?
  B739  MINE... >>B722
  B73B  MLI'S...SET EOF FARTHER INTO FILE
  B73D  MLI: SET EOF <BE70>
  B740  ERROR? >>B722
  B742  AND THEN TRY AGAIN TO SET MARK <B6D0>
  B745  ERROR? THEN I GIVE UP >>B722
  B747  BUFFER IS AT HIMEM
  B753  INDICATE OUTPUT "WRITE" FILE ACTIVE (BE45)
  B757   RETURN TO CALLER

B758 ********** "APPEND" COMMAND ***************************************

  B758  ---
  B759  LOOK UP NAME IN OPEN FILE LIST <B479>
  B75C  FOUND IT? >>B76A
  B75F  NO, OPEN IT FIRST <B392>
  B762  ERROR? >>B778
  B764  NO, REFNUM NON-ZERO? (BED0)
  B767  YES, OK >>B76B
  B769   ELSE, BREAK!!!
  B76A  ---
  B76B  REFNUM TO READ/WRITE PARM LIST (BED6)
  B76E  AND GET/SET LIST (BEC7)
  B771  DIR FILE? (BE47)
  B774  NO >>B77A

  B776  YES, "FILE LOCKED"
  B778  ---
  B779   EXIT TO CALLER

  B77A  PICK UP "L" VALUE (BE5F)
  B783  DID USER SPECIFY ONE?
  B785  YES... >>B78D
  B787  NO, USE FILE'S CURRENT "L" VALUE (BCA4)
  B78D  ---
  B792  COMPUTE REFNUM*32 FOR INDEX INTO
  B793  FILE NAME TABLE
  B798  SAVE CURRENT "L" VALUE IN OPEN FILE (BCFF)
  B79B  NAME TABLE AND IN CURRENT RECLEN (BCA4)
  B7A7  MLI: GET EOF <BE70>
```

```
BASIC Interpreter (BI) -- V1.0.1 -- 1 JAN 84    NEXT OBJECT ADDR: B7AA
-------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------

  B7AA  ERROR? >>B778
  B7AC  IS "L" VALUE < 2? (NO SPECIFIC "L") (BCA5)
  B7AF  NO >>B7B8
  B7B6  YES >>B7BD
  B7B8  NO, FORCE TO RECORD BOUNDARY <B7C0>
  B7BB  ERROR? >>B778
  B7BD  ELSE, GO SET EOF=MARK/OUTPUT FILE ACTIVE >>B73B

B7C0 *********** FORCE TO EVEN RECORD BOUNDARY **********************
          (FIND RECORD NUMBER OF THIS POSITION)

  B7C0  ---
  B7C2  COPY EOF TO ACCUM (BEC7)
  B7CB  CLEAR MSB'S (BCB2)
  B7D1  GET READY FOR A 24 BIT DIVIDE
  B7D3  DIVIDE EOF BY... <AB17>
  B7E0  RECORD LENGTH (BCA4)
  B7F5  ---
  B7FB  WAS THERE A REMAINDER? (BCB3)
  B7FF  NO, OK... >>B829
  B805  YES, CURRENT RECORD LEN LESS REMAINDER (BCB2)
  B812  PLUS OLD EOF MARK (BEC8)
  B81C  GIVES NEW EOF ON AN EVEN RECORD BOUNDARY (BEC9)
  B827  "RANGE ERROR" POSSIBLE IF OVERFLOW OCCURS
  B829   RETURN TO CALLER

B82A *********** GET FILE INFO ***************************************

  B82A  SET NUMBER OF PARMS (10)
  B82F  MLI CODE FOR GET FILE INFO
  B831  GO DO IT >>B848

B833 *********** SET FILE INFO ***************************************

  B833  MODIFIED TIME/DATE = 0
  B841  SET NUMBER OF PARMS (7)
  B846  MLI CODE FOR SET FILE INFO
  B848  EXIT THRU MLI: GET/SET FILE INFO >>BE70

B84B *********** BI I/O INDIRECTION VECTORS ***************************

  B84B  DOSOUT VECTOR >>BE38
  B84E  DOSIN VECTOR >>BE3A

B851 *********** STATE I/O VECTORS TABLE ******************************

  B851  IMMEDIATE MODE (STATE=0) CSWL/KSWL
  B855  DEFERRED MODE (STATE=4) CSWL/KSWL
  B859                (STATE=8) CSWL/KSWL
  B85D                (STATE=C) CSWL
```

```
BASIC Interpreter (BI) -- V1.0.1 -- 1 JAN 84    NEXT OBJECT ADDR: B85D
-------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------

B85F *********** SYSCTBL *********************************************
          LSB'S OF MLI CALL PARAMETER LISTS IN THE
          BI GLOBAL PAGE ($BEXX)

  B85F  CREATE: $A0  DESTROY: $AC  RENAME: $AF
  B862  SFI:    $B4  GFI:     $B4  ONLINE: $C6
  B865  SPFX:   $AC  GPFX:    $AC  OPEN:   $CB
  B868  NEWLINE:$D1  READ:    $D5  WRITE:  $D5
  B86B  CLOSE:  $DD  FLUSH:   $DD  SMARK:  $C6
  B86E  GMARK:  $C6  SEOF:    $C6  GEOF:   $C6
  B871  SBUF:   $C6  GBUF:    $C6

B873 *********** APPLESOFT TOKENS ************************************
          TOKENS REQUIRING SPECIAL ATTENTION HAVE
          THEIR MSB DFF AND ARE AN OFFSET FROM A
          JMP IN THE TRACE HANDLER IN THE BI

  B873  FIRST IS $80 (END)
  B87D   CALL
  B88D   TRACE, NOTRACE, NORMAL
  B891   INVERSE, FLASH
  B899   RESUME
  B89D   LET, IF
  B8AD   PRINT, LIST

B8B3 *********** COMMAND NAME TABLES **********************************
          OFFSETS TO LAST CHARCTER DF EACH COMMAND
          NAME IN THE COMMAND NAME TABLE BELOW.
          CDMMANDS ARE ARRANGED ACCORDING TO LENGTH
          WITH THREE BYTE NAMES FIRST.  IF THE MSB
          OF AN INDEX IS ON, THEN THIS IS THE LAST
          NAME OF THE GIVEN LENGTH (NEXT WILL BE
          ONE BYTE LONGER).

  B8B3  01 IN#      02 PR#      03 CAT
  B8B6  04 FRE      05 RUN      06 BRUN
  B8B9  07 EXEC     08 LOAD     09 LOCK
  B8BC  0A DPEN     0B READ     0C SAVE
  B8BF  0D BLOAD    0E BSAVE    0F CHAIN
  B8C2  10 CLOSE    11 FLUSH    12 NOMDN
  B8C5  13 STORE    14 WRITE    15 APPEND
  B8C8  16 CREATE   17 DELETE   18 PREFIX
  B8CB  19 RENAME   1A UNLOCK   1B VERIFY
  B8CE  1C CATALOG  1D RESTORE  1E POSITION

  B8D1  'BSAVERIFYBLOADELETECATALDGDPENWR'
  B8F1  'ITEXECREATEFRESTDRENAMEBRUNLOCKC'
  B911  'HAIN#FLUSHREADPOSITIDNDMONPR#PRE'
  B931  'FIXCLOSEAPPEND'
```

```
BASIC Interpreter (BI) -- V1.0.1 -- 1 JAN 84    NEXT OBJECT ADDR: B931
------------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
------------------------------------------------------------------------


B93F ********** COMMAND HANDLER ADDRESS TABLE **********************
          ADDRESSES OF THE COMMAND HANDLER ROUTINES
          FOR EACH COMMAND IN THE DRDER GIVEN ABOVE.

  B93F  (EXTERNAL)
  B941  IN#
  B943  PR#
  B945  CAT
  B947  FRE
  B949  RUN
  B94B  BRUN
  B94D  EXEC
  B94F  LOAD
  B951  LOCK
  B953  OPEN
  B955  READ
  B957  SAVE
  B959  BLOAD
  B95B  BSAVE
  B95D  CHAIN
  B95F  CLOSE
  B961  FLUSH
  B963  NOMON
  B965  STORE
  B967  WRITE
  B969  APPEND
  B96B  CREATE
  B96D  DELETE
  B96F  PREFIX
  B971  RENAME
  B973  UNLOCK
  B975  VERIFY
  B977  CATALOG
  B979  RESTORE
  B97B  POSITION
  B97D  "-" COMMAND

B97F ********** PERMITTED KEYWORDS FOR CMDS ************************
          TWO BYTES PER COMMAND IN THE ORDER ABOVE.
          EACH ENTRY HAS 16 BIT SETTINGS FOR THE
          PARAMETERS PERMITTED ON THAT COMMAND.
          8000 = FETCH PREFIX, PATHNAME OPTIONAL
          4000 = SLOT (FOR PR# OR IN#)
          2000 = DEFERRED COMMAND ONLY
          1000 = FILENAME IS OPTIONAL
          0800 = IF FILE NOT FOUND, CREATE IT
          0400 = "T" (FILE TYPE) PERMITTED
          0200 = PATHNAME2 (RENAME) PERMITTED
          0100 = PATHNAME1 EXPECTED
          0080 = "A" (ADDRESS) PERMITTED
```

```
BASIC Interpreter (BI) -- V1.0.1 -- 1 JAN 84    NEXT OBJECT ADDR: B97F
------------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
------------------------------------------------------------------------

          0040 = "B" (BYTE) PERMITTED
          0020 = "E" (END ADDRESS) PERMITTED
          0010 = "L" (LENGTH) PERMITTED
          0008 = "@" (LINE NO.) PERMITTED
          0004 = "S" AND/OR "D" (SLOT/DRIVE)
          0002 = "F" (FIELD) PERMITTED
          0001 = "R" (RECORD) PERMITTED
          ("V" IS IGNORED)

        C         P S D F N T P P A B E L @ S F R
        O         F L E E E   A A
        M         X O F O W   T T
        M         | T E P F   H H
        N         | | R T I   2 1
        D         | | | | L | | |
B97F    IN#       . X . . . . . . X . . . . . . .
B981    PR#       . X . . . . . . X . . . . . . .
B983    CAT       X . . X . . . X . . . . . X . .
B985    FRE       . . . . . . . . . . . . . . . .
B987    RUN       . . . X . . . X . . . . X X . .
B989    BRUN      . . . . . . . X X X X X . X . .
B98B    EXEC      . . . . . . . X . . . . . X X X
B98D    LOAD      . . . . . . . X . . . . . X . .
B98F    LOCK      . . . . . . . X . . . . . X . .
B991    OPEN      . . X . X X . X . . . X . X . .
B993    READ      . . X . . . . X . X . X . X X X
B995    SAVE      . . . . . X . . X . . . . . X . .
B997    BLOAD     . . . . . . X . X X X X X . X . .
B999    BSAVE     . . . . . X X . X X X X X . X . .
B99B    CHAIN     . . . . . . . . X . . . . X X . .
B99D    CLOSE     . . . . . . . . X . . . . . . . .
B99F    FLUSH     . . . . . . . . X . . . . . . . .
B9A1    NOMON     . . . . . . . . . . . . . . . . .
B9A3    STORE     . . . . . X . . X . . . . . X . .
B9A5    WRITE     . . X . . . . . X . X . . . . X X
B9A7    APPEND    . . X . X X . . X . . . X . X . .
B9A9    CREATE    . . . . . X X . X . . . . . X . .
B9AB    DELETE    . . . . . . . . X . . . . . X . .
B9AD    PREFIX    X . . X . . . . X . . . . . X . .
B9AF    RENAME    . . . . . . . X X . . . . . X . .
B9B1    UNLOCK    . . . . . . . . X . . . . . X . .
B9B3    VERIFY    . . . X . . . . X . . . . . X . .
B9B5    CATALOG   X . . X . X . . X . . . . . X . .
B9B7    RESTORE   . . . . . . . . X . . . . . X . .
B9B9    POSITION  . . X . . . . . X . . . . . . X X
B9BB    "-"       . . . . . . . . X . . . . . X . .
```

```
BASIC Interpreter (BI) -- V1.0.1 -- 1 JAN 84    NEXT OBJECT ADDR: B9BB
------------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
------------------------------------------------------------------------

B9BD *********** KEYWORD NAME TABLE ***********************************

  B9BD  'ABELSDFRV@'

B9C7 *********** KEYWORD BIT POSITION TABLE ****************************
          BIT POSITIONS IN PERMITTED PARMS TABLE
          FOR EACH KEYWORD IN THE ORDER GIVEN IN
          NAME TABLE.  "V" IS 00 (NOT USED)

  B9C7  ---

B9D1 *********** KEYWORD SIZE/OFFSET TABLE ****************************
          LOW 2 BITS - SIZE-1 OF VALUE IN BYTES
          HIGH 6 BITS- OFFSET TO LAST BYTE OF VALUE
                       FROM $BE58

  B9D1  A: 2 BYTES AT +1
  B9D2  B: 3 BYTES AT +4
  B9D3  E: 2 BYTES AT +6
  B9D4  L: 2 BYTES AT +8
  B9D5  S: 1 BYTE  AT +9
  B9D6  D: 1 BYTE  AT +A
  B9D7  F: 2 BYTES AT +C
  B9D8  R: 2 BYTES AT +E
  B9D9  V: 1 BYTE  AT +10  (IGNORED)
  B9DA  @: 2 BYTES AT +11

B9DB *********** FILE TYPES TABLES ***********************************
          FILE TYPE CODES, GIVEN IN INVERSE ORDER
          TO FILE TYPE NAMES WHICH FOLLOW.

  B9DB  $FF = "SYS"
  B9DC  $FE = "REL"
  B9DD  $FD = "VAR"
  B9DE  $FC = "BAS"
  B9DF  $FB = "IVR"
  B9E0  $FA = "INT"
  B9E1  $F0 = "CMD"
  B9E2  $0F = "DIR"
  B9E3  $06 = "BIN"
  B9E4  $04 = "TXT"
  B9E5  $EF = "PAS"
  B9E6  $1A = "AWP"
  B9E7  $1B = "ASP"
  B9E8  $19 = "ADB"
```

```
BASIC Interpreter (BI) -- V1.0.1 -- 1 JAN 84    NEXT OBJECT ADDR: B9E8
------------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
------------------------------------------------------------------------

  B9E9  ---
  B9E9  'ADBASPAWPPASTXTBINDIRCMDINTIVRBASVARRELSYS'

BA13 *********** MONTH TABLE ***************************************

  BA13  'JANFEBMARAPRMAYJUNJULAUGSEPOCTNOVDEC'
  BA37  '<NO          DATE>'

BA40 *********** MLIERTBL ****************************************
          MLI ERROR CODES WHICH HAVE BI EQUIVALENTS

  BA40  ---

BA53 *********** BIERTBL *****************************************
          BI EQUIVALENTS TO MLI ERROR CODES ABOVE
          (IF MLI CODE NOT FOUND, MAPS TO LAST CODE
          IN THIS TABLE, $08 "I/O ERRROR")

  BA53  ---

BA67 *********** INDEXS TO PACKED MESSAGES ***************************
          BY BI ERROR NUMBER

  BA67  ---

BA7B *********** COMMON LETTERS IN MESSAGES ***************************

  BA7B  ---
  BA7B  'ACDEFILMNORTU '

BA8A *********** LESS COMMON LETTERS *********************************

  BA8A  ---
  BA8B  'BGHKPSVWXY/().:'

BA9A *********** PACKED MESSAGES ***********************************

  BA9A   "COPYRIGHT APPLE COMPUTER"

  BAAA   " NAME ";TAB($10)
  BAAF  "TYPE  BLOCKS ";TAB($1E)
  BABA   "MODIFIED";TAB($2F)
  BAC0  "CREATED";TAB($40)
  BAC2
  BAC6   "ENDFILE SUBTYPE"
```

```
BASIC Interpreter (BI) -- V1.0.1 -- 1 JAN 84    NEXT OBJECT ADDR: BACE
-------------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------------

  BAD0  "BLOCKS FREE:";TAB($16)
  BADA   "BLOCKS USED:";TAB($2C)
  BAE5  "TOTAL BLOCKS:"

  BAEE   "RANGE ERROR"                    ERROR=$2

  BAF5  "NO DEVICE CONNECTED"            ERROR=$3

  BB00  "WRITE PROTECTED"                ERROR=$4

  BB09  "END OF DATA"                    ERROR=$5

  BB0F  "PATH NOT FOUND"                 ERROR=$6
  BB12   (NOT USED) --->                  ERROR=$7

  BB18  "I/O ERROR"                      ERROR=$8

  BB1E   "DISK FULL"                      ERROR=$9

  BB24  "FILE LOCKED"                    ERROR=$A

  BB2B  "INVALID PARAMETER"              ERROR=$B

  BB35  "RAM TOO LARGE"                  ERROR=$C

  BB42   "FILE TYPE MISMATCH"             ERROR=$D

  BB4E   "PROGRAM TOO LARGE"              ERROR=$E

  BB59  "NOT DIRECT COMMAND"             ERROR=$F

  BB63  "SYNTAX ERROR"                   ERROR=$10

  BB6B  "DIRECTORY FULL"                 ERROR=$11

  BB73  "FILE NOT OPEN"                  ERROR=$12

  BB7B  "DUPLICATE FILE NAME"            ERROR=$13

  BB86   "FILE BUSY"                      ERROR=$14

  BB8D  "FILE(S) STILL OPEN"             ERROR=$15

BB99 ********** PAUSE MESSAGE ****************************************

  BB99  LENGTH
  BB9A   (BLANK LINE)
  BB9E  BELL CHARACTER
  BB9F  'PLEASE       PRESS SPACE BAR '
```

```
BASIC Interpreter (BI) -- V1.0.1 -- 1 JAN 84    NEXT OBJECT ADDR: BBB6
-------------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
-------------------------------------------------------------------------

BBB6 ********** VARIABLES ****************************************

  BBB6  NUMBER OF PAGES TO ALLOCATE/FREE
  BBB7  NOT USED
  BBB8  TOP OF BUFFERS FOR GARBAGE COLLECTION
  BBB9  BOTTOM OF BUFFERS

BBBA ********** NOT USED ****************************************

  BBBA  NOT USED
  BBE4

BC7B ********** VARIABLES ****************************************

  BC7B  SAVED HIMEM VALUE DURING CHAIN LOAD
          ******** GARBAGE COLLECT MARKED GC: ****
  BC7C  GC: HIRANGE - WORKSAREASIZE
  BC7D  GC: WORKAREA MSB
  BC7E  GC: NUMBER OF PAGES IN WORKAREA
  BC7F  GC: LORANGE (START OF STRINGS TO COPY)
  BC80  GC: HIRANGE (END OF STRINGS TO COPY)
  BC81  ARRAYS START LSB
  BC82  ARRAYS ENDING MSB+1
  BC83  GC: START OF STRING AREA (ALSO PGM START)
  BC85  GC: END OF STRING AREA
  BC87  MSB ADJUST FACTOR FOR STRING POINTERS
  BC88  PAGE FOLLOWING BLOCK BUFFER
          ******** STORED VARIABLES FILE HEADER ***
  BC89  COMBINED LEN OF SIMPLE/ARRAY VARS
  BC8B  LEN OF SIMPLE VARS ONLY
  BC8D  HIMEM WHEN VARS WERE COMBINED
          ****************************************
  BC8E  POINTER TO COMBINED VARIABLES/STRINGS
  BC90  LENGTH OF COMBINED VARIABLES/STRINGS
  BC92  LENGTH OF STRINGS ONLY

  BC94   OPEN FILES' BUFFER MSBS
  BC9B  OPEN EXEC FILE BUFFER MSB

  BC9C   OPEN FILES' REFERENCE NUMBERS
  BCA3  OPEN EXEC FILE REFNUM

  BCA4  CURRENT RECORD LENGTH
  BCA6  NOT USED
  BCA9  CHARACTER TO FLUSH WHEN PARSING (BLANK)
  BCAA  MAXIMUM LENGTH TO PARSE
  BCAB  ADDRESS OF COMMAND HANDLING ROUTINE
  BCAD  SIZE OF KEYWORD VALUE -1 IN BYTES
  BCAE  OFFSET INTO KEYWORD PARMS TO LAST BYTE
  BCAF   GENERAL PURPOSE 4 BYTE ACCUMULATOR
```

```
BASIC Interpreter (BI) -- V1.0.1 -- 1 JAN 84    NEXT OBJECT ADDR: BCB3
------------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
------------------------------------------------------------------------

  BCB3  MONTH
  BCB4  DAY
  BCB5  YEAR
  BCB6  ERROR MSG LEN OR LINE LEN FOR CAT/CATALOG
  BCB7  ENTRY LENGTH IN DIRECTORY FILE
  BCB8  ENTRIES PER BLOCK IN DIRECTORY FILE
  BCB9  FILE COUNT FROM DIRECTORY FILE
  BCBB  DIRECTORY ENTRY NUMBER COUNTER

BCBC ********** PATHNAME 1 BUFFER **********************************

  BCBC  COMMAND OR PATH LENGTH
  BCBD  TXBUF (COMMAND OR PATHNAME STRING)
  BCFD  NOT USED

BCFE ********** OPEN FILE NAME TABLE ********************************
         (EACH ENTRY IS 32 BYTES LONG)
         (THERE ARE 8 ENTRIES)

  BCFE  FILE 0: LENGTH OF NAME
  BCFF  FILE 0: L VALUE LSB
  BD00  FILE 0: L VALUE MSB
  BD01  FILE 0: START OF NAME STRING
        (FILE NAME IS STORED BACKWARDS)
```

BASIC INTERPRETER GLOBAL PAGE

This page of memory is rigidly defined by the ProDOS BI. Fields given here will not move in later versions of ProDOS and may be referenced by external, user-written programs. Future additions to the global page may be made in areas which are marked "Not used".

ProDOS BI Global Page — NEXT OBJECT ADDRESS: BE00

ADDR	LABEL	CONTENTS
BE00-BE02	BI.ENTRY	JMP to WARMDOS (BI warmstart vector).
BE03-BE05	DOSCMD	JMP to SYNTAX (BI command line parse and execute).
BE06-BE08	EXTRNCMD	JMP to user-installed external command parser.
BE09-BE0B	ERROUT	JMP to BI error handler.
BE0C-BE0E	PRINTERR	JMP to BI error message print routine. Place error number in A-register.
BE0F	ERRCODE	ProDOS error code (also at $DE, Applesoft ONERR code).
BE10-BE1F	OUTVEC	Default output vector in monitor and for each slot (1-7).
BE20-BE2F	INVEC	Default input vector in monitor for each slot (1-7).
BE30-BE31	VECTOUT	Current output vector.
BE32-BE33	VECTIN	Current input vector.
BE34-BE35	VDOSIO	BI's output intercept address.
BE36-BE37		BI's input intercept address.
BE38-BE3B	VSYSIO	BI's internal redirection by STATE.
BE3C	DEFSLT	Default slot.
BE3D	DEFDRV	Default drive.
BE3E	PREGA	A-register savearea.
BE3F	PREGX	X-register savearea.
BE40	PREGY	Y-register savearea.
BE41	DTRACE	Applesoft TRACE is enabled flag (MSB on).
BE42	STATE	Current intercept state. 0 = immediate command mode. >0 = deferred.
BE43	EXACTV	EXEC file active flag (MSB on).
BE44	IFILACTV	READ file active flag (MSB on).
BE45	OFILACTV	WRITE file active flag (MSB on).
BE46	PFXACTV	PREFIX read active flag (MSB on).
BE47	DIRFLG	File being READ is a DIR file (MSB on).
BE48	EDIRFLG	End of directory flag (no longer used).
BE49	STRINGS	String space count used to determine when to garbage collect.
BE4A	TBUFPTR	Buffered WRITE data length.
BE4B	INPTR	Command line assembly length.
BE4C	CHRLAST	Previous output character (for recursion check).
BE4D	OPENCNT	Number of files open (not counting EXEC).
BE4E	YXFILE	EXEC file being closed flag (MSB on).
BE4F	CATFLAG	Line type to format next in DIR file READ.
BE50-BE51	XTRNADDR	External command handler address.
BE52	XLEN	Length of command name (less one).

ProDOS BI Global Page NEXT OBJECT ADDRESS: BE53

ADDR	LABEL	CONTENTS
BE53	XCNUM	Number of command: $00 = external $0A = OPEN $14 = WRITE $01 = IN# $0B = READ $15 = APPEND $02 = PR# $0C = SAVE $16 = CREATE $03 = CAT $0D = BLOAD $17 = DELETE $04 = FRE $0E = BSAVE $18 = PREFIX $05 = RUN $0F = CHAIN $19 = RENAME $06 = BRUN $10 = CLOSE $1A = UNLOCK $07 = EXEC $11 = FLUSH $1B = VERIFY $08 = LOAD $12 = NOMON $1C = CATALOG $09 = SAVE $13 = STORE $1D = RESTORE $1E = POSITION
BE54-BE55	PBITS	Permitted command operands bits: $8000 Prefix needed. Pathname optional. $4000 Slot number only (PR# or IN#). $2000 Deferred command. $1000 File name optional. $0800 If file does not exist, create it. $0400 T: file type permitted. $0200 Second file name required. $0100 First file name required. $0080 AD: address keyword permitted. $0040 B: byte offset permitted. $0020 E: ending address permitted. $0010 L: length permitted. $0008 @: line number permitted. $0004 S or D: slot/drive permitted. $0002 F: field permitted. $0001 R: record permitted. (V always permitted but ignored.)
BE56-BE57	FBITS	Operands found on command line. Same bit assignments as above.
BE58-BE59	VADDR	A keyword value.
BE5A-BE5C	VBYTE	B keyword value.
BE5D-BE5E	VENDA	E keyword value.
BE5F-BE60	VLNTH	L keyword value.
BE61	VSLOT	S keyword value.
BE62	VDRIV	D keyword value.
BE63-BE64	VFELD	F keyword value.
BE65-BE66	VRECD	R keyword value.
BE67	VVOLM	V keyword value (ignored).
BE68-BE69	VLINE	@ keyword value.
BE6A	VTYPE	T keyword value (in hex).
BE6B	VIOSLT	PR# or IN# slot number value.

ProDOS BI Global Page NEXT OBJECT ADDRESS: BE6C

ADDR	LABEL	CONTENTS
BE6C-BE6D	VPATH1	Primary pathname buffer (address of length byte).
BE6E-BE6F	VPATH2	Secondary pathname buffer (address of length byte).
BE70-BE84	GOSYSTEM	Call the MLI using the parameter tables which follow.
BE85	SYSCALL	MLI call number for this call.
BE86-BE87	SYSPARM	Address of MLI parameter list for this call.
BE88-BE8A		Return from MLI call.
BE8B-BE9E	BADCALL	MLI error return: translate error code to BI error number.
BE9F	BISPARE1	Not used.
BEA0-BEAB	SCREATE	CREATE parameter list.
BEAC-BEAE	SSGPRFX	GET_PREFIX, SET_PREFIX, DESTROY parameter list.
BEAF-BEB3	SRENAME	RENAME parameter list.
BEB4-BEC5	SSGINFO	GET_FILE_INFO, SET_FILE_INFO parameter list.
BEC6-BECA	SONLINE	ONLINE, SET_MARK, GET_MARK, SET_EOF, GET_EOF, SET_BUF, GET_BUF, QUIT parameter list.
BECB-BED0	SOPEN	OPEN parameter list.
BED1-BED4	SNEWLN	SET_NEWLINE parameter list.
BED5-BEDC	SREAD	READ, WRITE parameter list.
BEDD-BEDE	SCLOSE	CLOSE, FLUSH parameter list.
BEDF-BEF4	CCCSPARE	"COPYRIGHT APPLE, 1983"
BEF5-BEF7	GETBUFR	GETBUFR buffer allocation subroutine vector.
BEF8-BEFA	FREEBUFR	FREEBUFR buffer free subroutine vector.
BEFB		Original HIMEM MSB.
BEFC-BEFF		Not used.

ProDOS VERSION 1.Ø.2

In March, 1984, Apple began shipping Version 1.Ø.2 of ProDOS along with the Apple IIc. Version 1.Ø.2 is also the base for some of Apple's own software, such as AppleWorks. The differences between this version and its predecessor, Version 1.Ø.1, are minor. Except for the specific areas mentioned below, the description of Version 1.Ø.1 in this Supplement may be used for Version 1.Ø.2.

ProDOS Loader

Version 1.Ø.2 is identical to Version 1.Ø.1.

ProDOS Relocator

Replace the comments at the following addresses:

```
20A2:   YES, QUIT VECTOR -->$EEDB
21B8:   LEN = $1EDA
21C1:   TO  = $AF71
21C5:   FRM = $AF71
249A:   'PRODOS 1.Ø.2 15-FEB-84'
```

All other addresses and comments remain the same as Version 1.Ø.1.

ProDOS MLI (Kernel)

Replace the comments at the following addresses:

```
D19A:  Indicate error type 2
DE6D:  Stomp on $F3ØØ+$5E
```

At $E948, 12 bytes are added. This causes all addresses greater than $E947 and all references to those addresses to be increased by $ØC. For example, all references to $E948 in Version 1.Ø.1 become $E954 in Version 1.Ø.2. The 12-byte insertion is commented as follows:

```
E948:  Flush file; update directory <E71C>
E94B:  No error?  >>E954
E94D:  Error, return error code
```

ProDOS System Global Page

Version 1.Ø.2 is identical to Version 1.Ø.1.

ProDOS Quit Code

Version 1.Ø.2 is identical to Version 1.Ø.1. There is different data due to different uninitialized variables in a data area at the end of the Quit Code section, but this has no effect on the operation of the software.

ProDOS Disk II Device Driver

Minor changes to the beginning of the Disk II Device Driver caused the area from $F8ØØ to $F8F3 to change and added a routine at the end of the Version 1.Ø.1 code ($FEBE to $FED1). These two areas are described on the following pages. The rest of the Disk II Device Driver is identical to Version 1.Ø.1.

ProDOS IRQ Handler

Version 1.Ø.2 is identical to Version 1.Ø.1.

ProDOS BI Relocator

Version 1.Ø.2 is identical to Version 1.Ø.1.

ProDOS BASIC Interpreter (BI)

Version 1.Ø.2 is identical to Version 1.Ø.1.

ProDOS BI Global Page

Version 1.Ø.2 is identical to Version 1.Ø.1.

```
Disk II Device Driver -- V1.0.2 -- 15 FEB 84    NEXT OBJECT ADDR: F800
------------------------------------------------------------------------
ADDR    DESCRIPTIDN/CDNTENTS
------------------------------------------------------------------------
F800 ********** PRDDDS CDRE RDUTINES **********************************

  F800  Clear decimal mode
  F801  See if Block number is good <FEBE>
  F804  If not exit with error >>F834
  F806  Eight NDP's so code below will
  F807  fit up against Table at $F996
  F80E  Convert Block Number to a Track and Sector
  F810  ---
  F814  0000000T TTTTTABC
  F815           .     . >>F810
  F817         .       .
  F818       .         . >>F81C
  F81A  00TTTTTT 0000BC0A
  F81C  ---
  F820  Preserve Sector Number
  F821  Execute command <F838>
  F824  Restore Sector Number - Was prior action ok?
  F825  No, then exit >>F830
  F827  Increment Buffer Pointer
  F829  Increment Sector Number by 2 for rest of Block
  F82B  Execute command <F838>
  F82E  Decrement Buffer Pointer (to start of block)
  F830  Get error number (if any - 0 indicates no error) (FB58)
  F833  Return to caller

F834 ********** I/D ERRDR RDUTINE **************************************

  F834  Indicate "I/D Error"
  F836  Set Carry flag
  F837  Return to caller

F838 ********** MAIN CDDE **********************************************

  F838  Set recalibration count to 1
  F83D  Preserve sector number (FB57)
  F840  Get "Unitnum"   DSSS0000
  F842  Strip out Drive 0SSS0000
  F844  Preserve slot number
  F846  Check for slot change, turn off motor if so <FE9B>
  F849  See if motor is on <FCDA>
  F84C  Save test results
  F84F  Initialize counter for delay routine (FB70)
  F854  See if slot or drive has changed (FB59)
  F857  Update "current" unit number (FB59)
  F85A  Save test results
  F85B  Put drive number in Carry flag
  F85C  Turn motor on (C089)
  F862  Select appropriate drive (C08A)
  F865  Check test results - Same slot/drive?
  F866  Yes, then skip delay >>F872
```

```
Disk II Device Driver -- V1.0.2 -- 15 FEB 84    NEXT DBJECT ADDR: F829
------------------------------------------------------------------------
ADDR    DESCRIPTIDN/CONTENTS
------------------------------------------------------------------------

  F869  Wait for new Drive
  F86B  to come up to speed <FB85>
  F872  Is command a status request?
  F874  Yes, then do not move disk arm >>F87C
  F876  Get track number for current request (FB56)
  F879  And go there <F90C>
  F87C  Check test results - Was motor on?
  F87D  Yes, then skip delay >>F88E
  F87F  Wait for Drive to
  F881  come up to speed <FB85>
  F889  Is motor on yet? <FCDA>
  F88C  No, then exit with error >>F8EA
  F88E  Is command a "status" request?
  F890  Yes, then determine status >>F8FD
  F892  Is command a "read" request?
  F893  Yes, then continue on >>F898
  F895  Prepair data for write (prenibblize) <FDF0>
  F898  ---
  F89A  Initialize "retry" count at 64 (FB69)
  F89D  ---
  F89F  Read an address field - Good read? <FB98>
  F8A2  Yes, then continue on >>F8BE
  F8A4  Decrement "retry" count - More to try? (FB69)
  F8A7  Yes, then try again >>F89D
  F8A9  No, just in case indicate "I/O Error"
  F8AB  Decrement "recalibration" count - More to try? (FB6A)
  F8AE  No, then exit with error >>F8EA
  F8B0  Get "current" track (FB5A)
  F8B3  Preserve it
  F8B4  Double it and
  F8B5  add 16 to it for recalibration
  F8B7  Reinitialize Retry Count
  F8BC  Branch always taken >>F8CC
  F8C1  Was the right track found? (FB5A)
  F8C4  Yes, then continue on >>F8D5
  F8C6  Get "current" track (FB5A)
  F8C9  Preserve it
  F8CA  Get track we found
  F8CB  Double it
  F8CC  Put new value in Device Track Table <FCD3>
  F8CF  Get track we want
  F8D0  And go there <F90C>
  F8D3  Branch always taken >>F89D
  F8D8  Was the right sector found? (FB57)
  F8DB  No, then try again >>F8A4
  F8DF  Is command a "write" request?
  F8E0  Yes, then go do it >>F8F4
```

```
Disk II Device Driver -- V1.0.2 -- 15 FEB 84    NEXT OBJECT ADDR: F89F
------------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
------------------------------------------------------------------------

  F8E2  Read the data - Good read? <FBFD>
  F8E5  No, then try again >>F8A4
  F8E7  Indicate no errors
  F8E9  BNE Instruction, never taken
  F8EA  Indicate error
  F8EB  Preserve error number (FB58)
  F8EE  Get Slot
  F8F0  Turn motor off (C088)
  F8F3   Return to caller
```

```
Disk II Device Driver -- V1.0.2 -- 15 FEB 84    NEXT OBJECT ADDR: FEBB
------------------------------------------------------------------------
ADDR    DESCRIPTION/CONTENTS
------------------------------------------------------------------------

            ******* CHECK BLOCK NUMBER VALIDITY *

  FEBE  Get Block Number
  FEC2  Is Block Number good? (FB56)
  FEC5  Yes, if less than $100 >>FED0
  FEC8  No, if greater than or equal to $200 >>FECE
  FECC  No, if greater than or equal to $118 >>FED0
  FECE  Indicate error
  FECF  Return to caller
  FED0  All is well
  FED1  Return to caller
  FED2  Unused up to $FF00 >>002E
```

ERRATA TO BENEATH APPLE PRODOS (1st Printing, 1984)

Please make the following corrections to your copy of **Beneath Apple ProDOS:**

Page 3-16:

In the first paragraph starting on the page, the sentence should read "The data is dealt with in larger pieces (512 **bytes** vs. 256 **bytes**)...", not **512K** vs. **256K.**

Page 6-64:

The code for "GIVEN A PAGE NUMBER, SEE IF IT IS FREE" is incorrect. Replace it with:

```
BITMAP   EQU $BF58          SEE PAGE 8-6

         LDA #PAGE          GET PAGE NUMBER (MSB OF ADDR)
         JSR LOCATE         LOCATE ITS BIT IN BITMAP
         AND BITMAP,Y       IS IT ALLOCATED?
         BNE INUSE          YES, CAN'T TOUCH IT
         TXA                PUT BIT PATTERN IN ACCUM
         ORA BITMAP,Y       MARK THIS PAGE AS IN USE
         STA BITMAP,Y       UPDATE MAP
         ...                WE'VE GOT IT NOW

LOCATE   PHA                SAVE PAGE NUMBER
         AND #Ø7            ISOLATE BIT POSITION
         TAY                THIS IS INDEX INTO MASK TABLE
         LDX BITMASK,Y      PUT PROPER BIT PATTERN IN X
         PLA                RESTORE PAGE NUMBER
         LSR A              DIVIDE PAGE BY 8
         LSR A
         LSR A
         TAY                Y-REG IS OFFSET INTO BITMAP
         TXA                PUT BIT PATTERN IN ACCUM
         RTS                DONE

BITMASK  DFB $8Ø,$4Ø,$2Ø,$1Ø  BIT MASK PATTERNS
         DFB $Ø8,$Ø4,$Ø2,$Ø1
```

Page 7-26:

Modifying the ProDOS Disk II device driver to allow 320 blocks instead of the normal 280. The fourth command line should read:

```
520D:40
```

Modifying FILER to format 40 tracks instead of 35. The fourth command line should read:

```
4244:40
```

Page 8-6:

Under "Device Information", make the following changes:

```
BF10-BF11    DEVADR01    Slot 0 reserved.
...
BF26-BF27    DEVADR32    /RAM device driver address
                         (need extra 64K).
```

Page 8-7:

The wrong bit is indicated as the "expansion bit" in the MACHID byte. The first eight rows of that description should read:

```
00.. 0...  II
01.. 0...  II+
10.. 0...  IIe
11.. 0...  III emulation
00.. 1...  Future expansion
01.. 1...  Future expansion
10.. 1...  IIc
11.. 1...  Future expansion
```

Page B-8:

In the last paragraph, the sentence should read "A second way to use **an interpreted** language..." (not **a compiled** language).

Page D-1:

In the second paragraph, the sentence should read "Versions of the Disk Drive Controller Unit are now **used...**" (not **based**).

Reference Card, Panel 4

Under "SYSTEM GLOBAL PAGE FORMAT", replace the lines beginning BF00 and BF06 with the following two lines:

```
BF06    Jump to Date/Time Address
        (or RTS if no clock)
```

The description of BF10-11 should be changed to:

```
BF10-11  Slot 0 reserved
```

The description of BF26-27 should be changed to:

```
BF26-27  /RAM
```

Under the "MACHINE IDENTIFICATION BYTE", the second column of numbers should read:

```
0...
0...
0...
0...
1...
1...
1...
1...
```

Reference Card, Panel 9

The last entry for "MLI ERROR CODES" should be:

```
$5A  Bad vol. bit map
```

(not $58).

ORDERING FUTURE SUPPLEMENTS

New supplements will be published to reflect the changes made as ProDOS is updated. To order an updated supplement, mail the coupon on the next page directly to Quality Software (at the address on the coupon), along with a payment of $10.00 **plus** shipping and handling charges.* Your payment can be a check or bank draft in US dollars, or your VISA or MASTERCARD number and expiration date. California residents must add the appropriate sales tax (6 or 6.5%). No phone orders or CODs will be accepted.

footnote:
***SHIPPING & HANDLING CHARGES**
United States, Canada, and Mexico.....$ 2.50
All other countries (insured air mail).....$10.00